

# Lecture Notes in Computer Science

1746

Michael Walker (Ed.)

## Cryptography and Coding

7th IMA International Conference  
Cirencester, UK, December 1999  
Proceedings



Springer



*Berlin*  
*Heidelberg*  
*New York*  
*Barcelona*  
*Hong Kong*  
*London*  
*Milan*  
*Paris*  
*Singapore*  
*Tokyo*

Michael Walker (Ed.)

7th IMA International Conference  
Cirencester, UK, December 20-22, 1999  
Proceedings

## Series Editors

Gerhard Goos, Karlsruhe University, Germany  
Juris Hartmanis, Cornell University, NY, USA  
Jan van Leeuwen, Utrecht University, The Netherlands

## Volume Editor

Michael Walker  
Vodafone Limited  
The Courtyard, 2-4 London Road  
Newbury, Berkshire RG14 1JX, UK  
E-mail: mike.walker@vf.vodafone.co.uk

## Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

**Cryptography and coding** : ... IMA international conference ... ; proceedings. -  
5[?]-. - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;  
Milan ; Paris ; Singapore ; Tokyo : Springer, 1995[?]-  
(Lecture notes in computer science ; ...)

7. Cirencester, UK, December 20 - 22, 1999. - 1999  
(Lecture notes in computer science ; 1746)  
ISBN 3-540-66887-X

CR Subject Classification (1998): E.3-4, G.2.1, C.2, J.1

ISSN 0302-9743

ISBN 3-540-66887-X Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1999  
Printed in Germany

Typesetting: Camera-ready by author  
SPIN: 10750021 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

# Preface

fl

*future of cryptography and coding*



# Contents

Applications of Exponential Sums in Communications Theory .....	1
Some Applications of Bounds for Designs to the Cryptography .....	25
Further Results on the Relation Between Nonlinearity and Resiliency for Boolean Functions .....	35
Combinatorial Structure of Finite Fields with Two Dimensional Modulo Metrics .....	45
A New Method for Generating Sets of Orthogonal Sequences for a Synchronous CDMA System .....	56
New Self-Dual Codes over $GF(5)$ .....	63
Designs, Intersecting Families, and Weight of Boolean Functions .....	70
Coding Applications in Satellite Communication Systems .....	81
A Unified Code .....	84
Enhanced Image Coding for Noisy Channels .....	94
Perfectly Secure Authorization and Passive Identification for an Error Tolerant Biometric System .....	104

An Encoding Scheme for Dual Level Access to Broadcasting Networks .....	114
Photograph Signatures for the Protection of Identification Documents .....	119
An Overview of the Isoperimetric Method in Coding Theory .....	129
Rectangular Basis of a Linear Code .....	135
Graph Decoding of Array Error-Correcting Codes .....	144
Catastrophicity Test for Time-Varying Convolutional Encoders .....	153
Low Complexity Soft-Decision Sequential Decoding Using Hybrid Permutation for Reed-Solomon Codes .....	163
On Efficient Decoding of Alternant Codes over a Commutative Ring .....	173
Reduced Complexity Sliding Window BCJR Decoding Algorithms for Turbo Codes .....	179
Advanced Encryption Standard (AES) - An Update .....	185
The Piling-Up Lemma and Dependent Random Variables .....	186
A Cryptographic Application of Weil Descent .....	191
Edit Probability Correlation Attack on the Bilateral Stop/Go Generator .....	201

Look-Up Table Based Large Finite Field Multiplication in Memory Constrained Cryptosystems .....	213
On the Combined Fermat/Lucas Probable Prime Test .....	222
On the Cryptanalysis of Nonlinear Sequences .....	236
Securing Aeronautical Telecommunications .....	243
Tensor-Based Trapdoors for CVP and Their Application to Public Key Cryptography .....	244
Delegated Decryption .....	258
Fast and Space-Efficient Adaptive Arithmetic Coding .....	270
Robust Protocol for Generating Shared RSA Parameters .....	280
Some Soft-Decision Decoding Algorithms for Reed-Solomon Codes .....	290
Weaknesses in Shared RSA Key Generation Protocols .....	300
Digital Signature with Message Recovery and Authenticated Encryption (Signcryption) - A Comparison .....	307
Index .....	313

# Applications of Exponential Sums in Communications Theory

[Invited Paper]

h  
o  
- k o o  
o o ok - o  
o  
kp@hplb.hpl.hp.com

**Abstract.** o o o o o o o  
o o o o o o o  
o o o

## 1 Introduction

x h h  
h y  $\mathbb{F}$  h h  $\mathbb{F}$   
x y h  
x y x  
h h h x h -  
y h x x h h  
h h y h y  
h x  $\mathbb{F}$  h h -  
y h 9 h  
j h hy h h j h -  
j y h h- y h y  
h h h h h y -  
h h h h y x y -  
x h fly h h x x -  
xh h h y -  
y h h

h  
h h y h y h  
h h h h y h  
x h x h  
- - -  
h h x y y -  
h h h x h -  
- h y y x y -  
h y x fly  
x

$$\sum_{i=0}^{-1} 2^{mi}$$

$$\begin{array}{ccccccc}
& & o & o & o & & o & & o & o \\
& & & & y & h & & & & h \\
& & h & & & & h & & & \\
h & - & \mathbb{F}_{2^n} & & & \mathbb{F}_{2^n} & h & - & - & - & y \\
& & & & - & i^{n-1}(\cdot) & - & \mathbb{F}_{2^n} & & & \\
h & & & h & h & h & \text{additive characters} & \mathbb{F}_{2^n} & y & & y \\
h & & o & & h & h & \text{trivial} & h & & o & \mathbb{F}_{2^n} \\
- & \mathbb{F}_{2^n} & & h & - & h & & & & & \\
& & & & & & \sum_{-\mathbb{F}_{2^n}} & & & & \\
& & h & - & & & h & & & & \\
& & & - & & x & & x & - & h & y \\
& & \mathbb{F}_{2^n} & h & & h & & h & - & & - \\
& & & & & & y & & & & \\
& & & & & - & & - & & & \\
h & & h & & h & \text{multiplicative characters} & \mathbb{F}_{2^n} & h & y & h & - \\
h & & \mathbb{F}_{2^n} & - & h & o & h & \text{trivial} & & & \\
& & h & & h & & h & & h & h & h \\
& x & & h & & h & & h & & h & y \\
& & & & h & - & h & & & & \\
& y & & & & & & & & & \\
- & 2- & & \mathbb{F}_{2^n} & & h & & h & - & - & - \\
& & - & h & & y & & & & & \\
& & - & - & 1 & 3 & 3 & - & 2 & -1 & 2 & -1 & - & \mathbb{F}_{2^n} & - \\
h & - & - & & h & - & y & & & & & \\
& & & 1 & & 1 & & 1 & & 2^n-2 & & \\
& & \mathcal{C} & y & & & & & & & & \\
& & & & \mathcal{C} & - & - & - & - & & & \\
h & & \mathcal{C} & & y & & 2^n-2 & \mathbb{F}_{2^n} & h & - & y & y & h \\
h & & -z & & & & & & & y & & & \\
& & & h & & h & y & h & & h & h & \mathcal{C} & \mathbb{F}_2 & h & & \\
& & h & - & 2 & -1 & - & - & - & \mathbb{F}_{2^n} & - & & h & & y \\
x & y & & h & & y & & y & h & h & h & h & & y & &
\end{array}$$







### 3.2 Hybrid Exponential Sums

**Definition 1** Let  $\chi$  be an additive character and  $\psi$  a multiplicative character of  $\mathbb{F}_{2^n}$ . Then the Gaussian sum  $G(\chi, \psi)$  is defined by

**Result 2** *Let  $\chi$  be a non-trivial additive character and  $\psi$  a non-trivial multiplicative character of  $\mathbb{F}_{2^n}$ . Then*

**Result 3** Let  $\chi$  be a non-trivial multiplicative character of  $\mathbb{F}_{2^n}$  of order  $m$  with  $\gcd(m, 2^n - 1) = 1$ . Let  $\psi$  be a non-trivial additive character of  $\mathbb{F}_{2^n}$ . Let  $\alpha \in \mathbb{F}_{2^n}^*$  have distinct roots and  $\beta \in \mathbb{F}_{2^n}^*$  have degree  $n$ . Suppose that  $\alpha \neq \beta$  and that  $n$  is odd. Then

$$\left| \sum_{-\mathbb{F}_{2^n}} \right| \quad \quad \quad - \quad \quad - \quad \quad 2$$

h h h y  
h y h y  
h h h h h h h h  
h h h z h h h h

[illegible]

**Theorem 4.** Suppose  $\frac{1}{2} - \frac{1}{2^{k+1}} \leq \frac{1}{2} - \frac{1}{2^{k+1}}$ . Then the minimum Hamming distance of  $\mathcal{C}$  is at least  $\frac{1}{2} - \frac{1}{2^{k+1}}$ .

# h

## 5 Application: Sequence Sets with Low Periodic Correlations

$$\begin{array}{cccccccccccc}
 h & & & & & & y & & & & & - \\
 & & - & & & & - & & & & & \\
 & & y & h & & & & & & & & \\
 & & h & & y & & h & & & & & \\
 & & h & & & & & & h & & & \\
 y & & & & h & & & h & & h & & - \\
 & & & & & & h & & & & & \\
 h & - & & & h & & & & y & h & h & \\
 x & & & & & h & & & & & & \\
 & & - & & h & & & & & & & 
 \end{array}$$

### 5.1 Periodic Correlation Functions

$$\begin{array}{ccccccc}
 & 0 & 1 & 2 & & 0 & 1 & 2 & & x- \\
 & y & h & h & & + & & + & & - \\
 h & \text{periodic cross-correlation} & & & & & h & & - & 
 \end{array}$$

$$\sum_{\tau=0}^{-1} \overline{-+}$$

$$\begin{array}{ccccccc}
 - & h & \text{periodic cross-correlation function} & & h & & \\
 & h & y & h & & h & \\
 h & \text{periodic auto-correlation} & & h & - & & 
 \end{array}$$

$$\begin{array}{ccccccc}
 h & & - & & h & - & y & h \\
 h & & h & & h & - & h & \\
 \sum_{\tau=0}^{-1} - \underline{2} & & y & & h & & h & \\
 & & & & h & - & & -z & h \\
 y & & \text{non-trivial auto-correlations} & & & & & & 
 \end{array}$$

### 5.2 A Simplified Model for CDMA Communications

$$\begin{array}{ccccccc}
 x & & & & & & y & h & \\
 h & & & & h & & h & & h \\
 y & h & & h & & h & & & h \\
 & & h & j & & y & h & & \\
 & & & & & h & & & \\
 h & & \text{spreading code} & h & h & & j & & x- \\
 & & & & h & & & & 
 \end{array}$$

$$\begin{aligned}
& \text{---} \quad \text{---} \quad \text{---} \qquad y \qquad h \qquad \text{---} \quad j \\
& h \\
& \qquad \text{---} \quad j \quad 0 \quad \text{---} \quad j \quad 1 \quad \text{---} \quad j \quad 2 \\
& h \qquad h \qquad \qquad \text{---} \quad \text{---} \quad \text{---} \qquad h \qquad spread \quad y \quad h \\
& h \qquad \qquad \qquad y \qquad \qquad \qquad 0 \quad 1 \quad 2 \qquad h \\
& \sum_{=0}^{-1} - \quad j \quad + \quad j \\
& h \quad delay \qquad \qquad h \qquad \qquad h \qquad \qquad h \qquad \text{---} \\
& h \qquad \qquad h \qquad h \qquad y \quad h \qquad h \qquad h \qquad h \\
& \qquad h \qquad h \qquad h \qquad h \qquad h \qquad h \\
& h \qquad \qquad h \qquad h \quad \text{---} \qquad h \qquad \qquad h \\
& \sum_{=0}^{-1} \left( \sum_{=0}^{-1} - \quad j \quad + \quad j \right) \text{---} \quad + \\
& \sum_{=0}^{-1} \left( \sum_{=0}^{-1} - \quad j \quad + \quad j \text{---} \quad + \right) \\
& \text{---} \quad \ell \qquad \text{---} \qquad \sum_{=} - \quad j \qquad \text{---} \\
& h \qquad h \quad \text{---} \qquad h \qquad \qquad h \qquad \text{---} \\
& y \qquad \text{---} \qquad \qquad h \qquad \qquad y \qquad \qquad y \quad y \\
& \text{---} \quad \ell \qquad h \qquad h \quad x \qquad \qquad h \qquad \qquad h \\
& \qquad h \qquad h \qquad \qquad h \quad \text{---} \quad h \qquad h \qquad h \\
& h \qquad \qquad \qquad h \qquad y \qquad \qquad h \qquad \qquad h \\
& h \qquad y \qquad h \qquad h \qquad h \qquad h \qquad y \qquad \text{---} \\
& \qquad h \qquad \text{---} \quad \ell \qquad h \qquad \qquad y \quad h \quad h \quad \text{---} \\
& \qquad h \qquad \qquad \qquad x \qquad \qquad y \quad h \qquad h \\
& h \qquad \text{---} \quad \text{---} \quad \text{---} \quad \text{---} \qquad h \\
& \text{---} \qquad \qquad x \text{---} \\
& \max \text{---} \qquad \text{---} \quad 1 \text{---}^x \quad \text{---} \quad \text{---}
\end{aligned}$$



o

$$\begin{array}{ccccccc} h & & h & & h & & h \\ - & & - & \square & & \gamma & - \\ & & & - & & & \\ & & h & & h & & h \\ & & & & & & - \\ & & & & & & h \\ & & & & & & \\ & & & & & & h \end{array}$$

5.4 Sequence Sets from  $m$ -Sequences

$$\begin{array}{ccccccc} x & & h & - & - & & y & & h \\ & & h & - & & & - & & \\ & & & y & x & & h & - & x \\ & & & & & & & & \\ & & & h & & & & & \\ & & & & & & \mathbb{F}_{2^n} & & - \\ & 0 & 1 & & y & & h & & \\ & & & & h & & - & & \\ & & & & & & & & - \\ & & & & & & & & - \\ & & & & h & & - & & \\ & & & & & & & & \\ & & & & & & \sum_{i=0}^{2^n-2} - & \frac{n}{i} ( - d i ) & - - & \frac{n}{i} ( - i + \tau ) \\ & & & & & & \sum_{-\mathbb{F}_{2^n}} - & \frac{n}{i} ( + d ) & & h \\ & & & & & & - & \sum_{-\mathbb{F}_{2^n}} - & \frac{n}{i} ( + d ) \\ & & & & & & & & \\ & & & & & & y & & h \\ & - & y - & & h & h & y & h & - & h & y & - & h \\ & & & & h & & z & & h & x & & \\ & y & & x & & & & & h & & \\ & & & & & & y & h & h & & \max - \\ & & & & h & & & & y & h & h \\ & & & & & & x & y & & & \\ & & & & & & & & & & \\ & h & & x & & h & & & & & -( + 2 ) 2 - & h & h \\ & & & & & & & & & & & \\ & h & & y & & - & & & & & & \\ & & & & & & & & & & h & h & h \\ & & & & & & & & & & 9 & h & \\ & & & & & & & & & & & & binary \end{array}$$
$$\begin{array}{ccccccc} & & & h & y & & h & & h \\ & & & & & & & & h & h \\ y & & & h & & & & & h & h \\ & & & x & & & h & & h & y & h \\ \mathbb{F}_{2^n} & & \mathbb{F}_2 & h & & & & & y & & \\ & & & & & & & & & & \\ & & & \mathbb{F}_2 & h & & 1 & & y & h & h & x \end{array}$$

h y x h h y h rank h y h h x y h h h h h h j 2 - y h y h y h y

### 5.5 Sequence Sets from Dual BCH Codes

[illegible]



$$\begin{array}{ccccc} \circ & \circ & \circ & & \circ & \circ & \circ \end{array}$$

$$\begin{array}{ccc} & x & y & h \end{array}$$

$$\begin{array}{ccccccc} & & & y & & & \\ - & & h & h & \textit{partial cross-correlation} & & \\ & h & & h & & & h \\ h & & & & y & & \end{array}$$

$$\sum_{=0}^{-1} + \overline{-++}$$

$$\begin{array}{ccc} y & h & \textit{partial auto-correlation} & y \end{array}$$

$$\begin{array}{ccccccccccc} & h & h & & h & & & h & & & \\ y & & h & & y & & y & h & & y & h \\ & y & & & y & & & 9 & & y & h \\ & & y & y & & & y & h & & 9 & h \\ & & & & & & y & h & & & h \end{array}$$

$$\begin{array}{ccccccc} h & h & h & x & & y & h \\ & & & & h & - & \\ & & & & & & h & h & h & & h & - \end{array}$$

$$\begin{array}{ccccccc} h & & & x & & h & h \\ & & h & & & & h & h & h \\ & & & h & x & & h & & h \end{array}$$

$$\begin{array}{ccccccc} - & & & h & & y & h \\ & & & & & & & y & h & h & - \end{array}$$

$$\begin{array}{ccccccc} h & x & & h & & y & h \\ h & & x & h & y & x & \\ h & & & h & & & h & h & y \end{array}$$

$$\begin{array}{ccccccc} h & h & & h & y & - & h & h & y \\ h & h & & & h & y & & & y \\ 9 & & & & & & & & \end{array}$$

$$\begin{array}{ccc} & \left\{ \begin{array}{c} h \\ h \end{array} \right. & - \end{array} \qquad \qquad \qquad - \mathbf{Z}$$

$$\sum_{=0}^{-1} - \overline{-+}$$







$$\begin{array}{ccccccc}
& & o & o & o & & o & o & o \\
& & & & & & & & \\
& & y & & h & h & h y & x & h & - \\
& & & & h & h & h & & & \\
& & y & & & & h & & h & -z \\
& & & & y & & y & h & & \\
& & & & & & & y & & h \\
h & & & & & & & & & \\
& & h & & -z & & h & h & & - \\
\mathcal{C} & & h & & & & & & &
\end{array}$$

$$\begin{array}{ccccccc}
& & 1 & & - & & \\
& & & & & & \\
h & & y & & - & \mathbb{F}_{2^n} & \mathbb{F}_{2^n} \\
h & & y & & h & &
\end{array}$$

$$\sum_{j=0}^{-1} - \quad {}^n_1((\quad^j))$$

$$\begin{array}{ccccccc}
& & x_{-1} - & \underline{2} & h & & (2 \quad) \\
- h & & y & h & & &
\end{array}$$

$$(2 \quad) \quad \sum_{j=0}^{-1} - \quad {}^n_1((\quad^j)) \quad (2 \quad) \quad \sum_{- \mathbb{F}_{2^n}} 1$$

$$\begin{array}{ccccccc}
h y & x & h & y & h & - & h \\
& h & & h & & & \\
y & y & & & & &
\end{array}$$

$$\begin{array}{ccccccc}
& & - & (2 \quad) & - & - & 2 \\
& & & & & &
\end{array}$$

$$\begin{array}{ccccccc}
h & h & h & & & &
\end{array}$$

$$- \quad - \quad \left| \sum_{j=0}^{-1} - \quad {}^n_1((\quad^j)) \right| \quad \left| - \quad \sum_{- \mathbb{F}_{2^n}} 1 \quad \right| \quad - \quad - \quad 2$$

$$\begin{array}{ccccccc}
& h & h & h & - h & y & h & y & - \\
& h & & h & h & - & 2 & & \\
& h & h & h & h & - h & y & & \\
h & & h & & h & & & & \\
& y & h & h & h & & & & \\
& - & y & h & - h & y & - & &
\end{array}$$

**Lemma 4** [46] *Let  $\gamma$  be a degree  $\leq 2$  polynomial and write  $\gamma = \gamma_0 + \gamma_1 x + \gamma_2 x^2$ . Then*

$$\sum_{j=0}^{-1} - \quad \left( - \quad \right)_{0-} x - \gamma -$$

h h h h h x

[illegible]

## 8 Further Applications and Literature

elliptic curves

o o o o o o o  
 h - y- -  
 -1 h h *reciprocal* - h h h  
 h h h h h h h  
 h h - - - h h h h y h  
 h h h h h h y  
 h h h h h x  
 x h h y 9  
 x 9 h h y h  
 x *rings* h h h y  
 h z- h y h hy  
 x h h h y y  
 - y h h h hy  
 y h y y h h x  
 h h h h

# References

o o o *EBU Review* o o  
 o o *IEEE Commun. Magazine* o o  
 o *IEEE Trans. Inform. Theory* - o - k o o  
 o o o *Duke Math. J.*  
 o o - o -  
 o *IEE Colloquium on 'High Speed Access Technology and Services, Including Video-on-Demand'*  
 o o o o o -  
 o *IEEE Trans. Commun.*  
 k- o- o o o o o -  
 - o *IEEE Trans. Inform. Theory* o  
 o  
 o *Publ. Math. IHES*

o  
 o Spread Spectrum Systems with Commercial Applications (3rd edition)  
 o k  
 o o GF<sup>n</sup>  
 IEEE Trans. Inform. Theory -  
 o o GF<sup>n</sup> o  
 Information and Computation o  
 o k o o Amer. J. Math.  
 Sequence design for communications applications o  
 o k  
 k o o o k  
 Trans. of IEICE  
 o o IEEE  
 Trans. Inform. Theory -  
 o - o o o  
 IEEE Trans. Inform. Theory -  
 o o q IEEE Trans.  
 Inform. Theory -  
 o k o o  
 Z<sub>4</sub>- o o k o o IEEE Trans.  
 Inform. Theory -  
 o o o  
 Discrete. Appl. Math.  
 o oo o o o o  
 o o o m- preprint  
 o o Acta Applicandae  
 Mathematicae  
 o A Classical Introduction to Modern Number Theory  
 (2nd edition), Graduate Texts in Mathematics Vol. 84  
 k Finite Fields — Structure and Arithmetics -  
 d o o o o  
 IEEE Trans. Inform. Theory -  
 k o o -  
 o o o IEEE Trans. Inform. Theory  
 -  
 k o  
 o o o IEEE Trans. Inform. Theory -  
 o o o  
 IEEE Trans. Inform. Theory -  
 o k oo o  
 Comptes Rendu Academie Science Paris  
 o o o o o o  
 IEEE Trans. Inform. Theory -  
 o o o o o  
 o o o Soviet Math. Dokl.

Introduction to finite fields and their applications  
 (2nd Edition)  
 Finite Fields  
 AAECC  
 The Theory of Error-Correcting Codes  
 Finite fields for computer scientists and engineers  
 Algebraic Curves over Finite Fields  
 IEEE Trans. Inform. Theory  
 Trans. Inform. Theory  
 Finite Fields and their Applications  
 IEEE Trans. Inform. Theory  
 IEEE Trans. Inform. Theory  
 IEEE Trans. Inform. Theory  
 IEEE Transactions on Information Theory  
 IEEE Transactions on Information Theory  
 IEEE Transactions on Information Theory  
 Hewlett-Packard Laboratories Technical Report HPL-1999-51, submitted  
<http://www.hpl.hp.com/techreports/1999/HPL-1999-51.html>  
 Handbook of Coding Theory Vols. I & II  
 Proc. of Conf. on Information Sciences and Systems  
 IEEE Trans. Commun.  
 IEE Proc. (F)  
 IEEE Trans. Inform. Theory

o

- o o o o o o o  
IEEE Trans. Inform. Theory -

o - o o o o o o

Proc. IEEE

Equations Over Finite Fields — An Elementary Approach. Lecture  
Notes in Mathematics, Vol. 536

o o o o o

o o o o o Applied Algebra, Algebraic Algorithms  
and Error-Correcting Codes (AAECC-10) o o

-

o IEEE Trans. Inform. Theory

- o  
o Number Theory in Science and Communication (3rd edition)

o o o

o o o o o o o q-  
IEEE Trans. Inform. Theory -

ko o o o Soviet Math. Dokl.

o o Spread Spectrum Com-

munications, Vol. 1 o o k  
o Algebraic Function Fields and Codes - o k

.. .. o o o o o o

IEEE Trans. Inform. Theory. -

.. .. o o o o o o

o k o o o k o

o

o o  $\mathbb{Z}_4$  IEEE Trans. Inform. Theory -

o o Elements of Number Theory o o k

Sur les courbes algébriques et les variétés qui s'en déduisent, Actualités  
Sci. et Ind. no. 1041

o o o o o o IEEE Trans.

Inform. Theory -

# Some Applications of Bounds for Designs to the Cryptography

l k \* n n l k

Department of Mathematics and Informatics  
Veliko Tarnovo University  
5000 Veliko Tarnovo, Bulgaria  
svetla\_venci@hotmail.com

**Abstract.** Recent years have seen numerous examples where designs play an important role in the study of such topics in cryptography as secrecy and authentication codes, secret sharing schemes, correlation-immune and resilient functions. In this paper we give applications of some methods and results from the design theory, especially bounding the optimal size of the designs and codes, to cryptography. We give a new bound for the parameter  $t$ , when  $(n, T, t)$ -resilient functions and correlation-immune functions of order  $t$  exist. In the last section we present analogous bound for the parameter  $N$  of  $T$ -wise independent  $t$ -resilient function.

## 1 Introduction

minimal distance  $n$   $n$   $code$   $design$   $n$   $x y$   $n$   $n$   $l$   
 $x, y$   $C, x=y$   $x y$   $n$   $l$   $n$   
 $l$   $x y$   $n x y$   $n$   $n$   
 $n$   $n$   $n$   $n$   $n$   $l$   
 $n$   $n$   $ll$   $n$   $n$   
 $n$   $l$   $n$   $l$   $n$   $n$   $n$   $n$   $n$   
 $n$   $l$

**Definition 1.1** A set will be referred to as a  $t$ -design in  $\Omega$  with respect to the substitution  $\sigma$  if for any polynomial  $f$  in a real  $\Omega$  of degree at most  $t$ ,

$$\int \int f(x, y) \, dx \, dy = \frac{1}{2} \sum_{x, y \in C} f(x, y)$$

\* Supported by a junior research fellowship of the Katholieke Universiteit Leuven, Belgium.



$$\begin{array}{cccccccccccc} n & n & n & & nn & n & & n & & & l & n \\ & & n & & n & & & n & & l & n & n \\ l & n & & n & n & n & & n & l & & n & n \\ & n & & & & & & n & n & n & l & n \end{array}$$

## 2 Improvement of the Delsarte Bound for Orthogonal Arrays and Combinatorial Designs

$$\begin{array}{cccccccccccc} & & l & & n & & n & & n & & & \\ n & n & l & & & & l & & n & & n & \\ n & n & & n & n & n & l & n & l & n & l & \\ l & n & & & & n & & & l & & l & \\ ll & n & n & n & n & ll & n & & & & & \\ & & n & & n & n & & & n & l & & \\ & & & l & n & l & i & & & & - & \\ ll & zonal\ spherical\ functions & & & l & n & l & & & & - & ll \\ \text{antipodal} & & & n & x & - & - & & n & \bar{x} & - & - \\ n & n & y & - & - & & x & y & & \bar{x} & y & \\ x & y & - & & & n & & n & & ll & n & - \\ & n & - & n & & & l & n & n & n & n & n \\ & & n & i & & i & & & i & & - & \\ \sum_{i=0}^s & i & & & - & - & - & n & n & ll & \text{adjacent} & \\ & n & l & l & n & l & \frac{a,b}{k} & & - & a & b & \\ n & l & & & & - & - & a_{,1} & - & b_{,1} & n & n & c^{a,b} \\ n & \frac{a,b}{k} & & & & n & & l & & i & c_i & - & \frac{i}{a_{i+1,i+1}} \\ i & i+1 & C_i & i-1 & & - & -1 & -1 & & i & \frac{a_{i,i}}{a_{i+1,i+1}} & & \\ c_i & \frac{r_{i-1}m_{i-1}}{r_i} & n & -1 & - & 0 & - & n & \frac{a,b}{k} & \frac{a,b}{k} & \sum_{i=0}^k \frac{a,b}{k,i} & i & \\ & l & n & l & \frac{a,b}{k} & ll & n & n & n & \frac{a,b}{k} & & & \\ \frac{a,b}{i} & \frac{\frac{a,b}{a_{i,i-1}}}{\frac{a,b}{a_{i,i}}} & \frac{a,b}{i} & \frac{\frac{a,b}{a_{i,i-2}}}{\frac{a,b}{a_{i,i}}} & \frac{a,b}{i} & \frac{\frac{a,b}{a_{i,i}}}{\frac{a,b}{a_{i,i}}} & & & & & & & \\ & l & n & & n & n & n & n & n & n & n & n & \\ ll & n & & & & & & & & & & & \end{array}$$

**Theorem 2.1.** Let  $\mathcal{C}$  be an  $(n, k, \lambda)$ -code (reps.  $t$ -design) and let  $f$  be a real non-zero polynomial such that

- (A1)  $f$  is a  $t$ -th degree polynomial for  $t \leq k$ , (resp. (B1)  $f$  is a  $t$ -th degree polynomial for  $t \leq k$ ),  
 (A2) the coefficients in the ZSF expansion  $f = \sum_{i=0}^k f_i \phi_i$  satisfy  $f_0 = 0$ ,  $f_i = 0$  for  $i > k$ . (resp. (B2) the coefficients in the ZSF expansion  $f = \sum_{i=0}^k f_i \phi_i$  satisfy  $f_0 = 0$ ,  $f_i = 0$  for  $i > k$ .)

Then,  $\Omega(f) \geq f_0$  (resp.  $\Omega(f) \leq f_0$ ).

$$\begin{array}{cc} \text{(B1)} & \text{(B2)} \\ n & n \\ - & - \\ \Omega & \Omega \end{array}$$





n n ll n

**Theorem 2.7.** [5] Let  $C$  be an  $[n, k]$ -code (reps.  $t$ -design) and let  $f = \sum_{i=0}^n f_i x^i$  be a real non-zero polynomial such that

(C1)  $f \quad , \quad f \quad - \quad for \quad ,$   
 (resp. (D1)  $f \quad , \quad f \quad - \quad for \quad ),$

(C2)  $f_0 \leq f_i$  for  $i = 1, \dots, n$ .  
 (resp. (D2)  $f_0 \geq f_i$  for  $i = 1, \dots, n$ .)

Then,  $-\Omega f$  (resp.  $-x\Omega f$ ), where  $\Omega f = f - f_0$ .

$$\begin{array}{ccccccc} & n & A_v^* & -\Omega f & l_n & l_f & n \\ n & (C1), (C2) - & v^* & x-\Omega f & l_n & l_f & n \\ n & (D1), (D2) - & n & v^{**} & x-\Omega f & l_n & l_f \\ n & D1 \quad D2 & n & f - & - & & \end{array}$$

**Theorem 2.8.** [2,5] For any integers  $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z$ ,

$$A_v^* - n$$
$$\tau \quad \text{---} \quad \text{---} \quad \text{---} \quad n-d+1$$

$\frac{n}{v}$

$$\begin{array}{ccccccc} & & & & n & & \\ n & & & & & & \\ v & - & - & - & - & - & - \\ & & & & n & & \\ & & & & v & & - \end{array}$$
$$ll \quad \begin{matrix} n & l & & l & & n & & n \\ tight & & n & n & perfect & & l \\ & n & & l & n & & n & - \frac{n}{v} \end{matrix} \quad \text{Levenshtein}$$

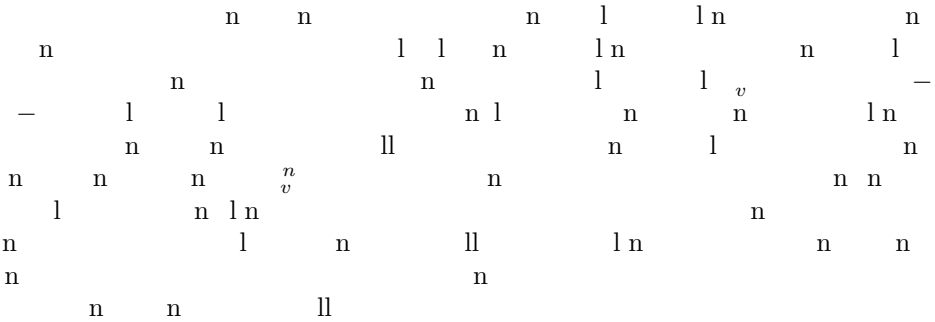
n

$$\begin{array}{ccccccc} & & n & & & & n \\ \hline & & & & - & - & - \\ n & & & & & & v \\ v & & & & & & \end{array}$$

**Theorem 2.9.** [12] For any code  $\left[ \begin{smallmatrix} n \\ v \end{smallmatrix} \right]$

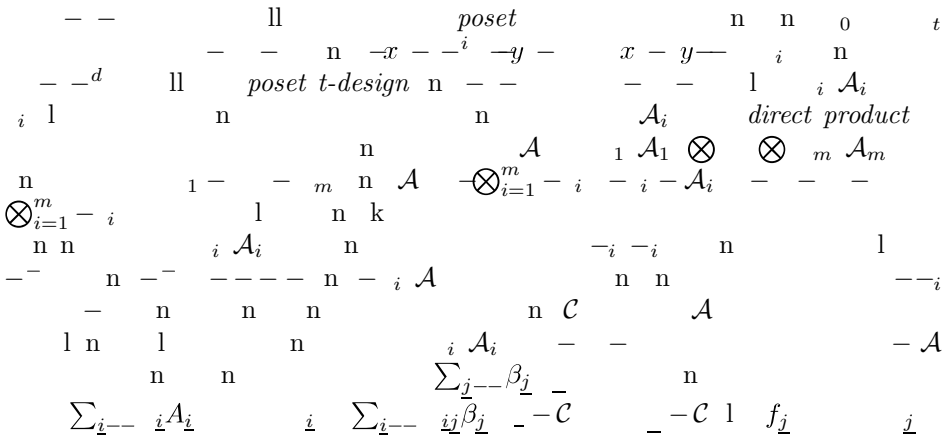
$$\begin{array}{ccccccc} & & & & n & & \\ n & & & & & & \\ v & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ & & & & n & & \\ & & & & v & & \text{---} \end{array}$$





**Theorem 3.6.** [1,4] Suppose there exists a  $t$ -resilient function and  $\mathcal{A}$ . Then  $\frac{2^{T-1}n}{2^T-1} \leq \frac{2^{T-2}(n+1)}{2^T-1}$ .

#### 4 Designs in Product Association Schemes. Maximum Independent Resilient System of Functions



**Theorem 4.1.** [9] Let  $\mathcal{A}$  be the product of  $t$ -polynomial association schemes  $\mathcal{A}_i$ . Let  $\mathcal{C}$  be a downset in  $\mathcal{C}$  and let  $\mathcal{C}$  be a Delsarte  $t$ -design. Consider the matrices satisfying the conditions

- (i)  $\beta_{\mathcal{C}}$  is non-negative matrix;
- (ii)  $\beta_{\mathcal{C}} = 0$  for  $\mathcal{C} \neq \mathcal{C}$ ;
- (iii)  $\beta_0 = 1$ .

Then, the lower bound on the size of a  $t$ -design is  $\frac{1}{\beta_0}$ .

**Theorem 4.2. (Delsarte bound)**[9] Let  $\mathcal{A}$  be the product of  $t$ -polynomial association schemes  $\mathcal{A}_i$ . Let  $\mathcal{C}$  be a downset in  $\mathcal{C}$  and let  $\mathcal{C}$  be a Delsarte  $t$ -design. If  $\mathcal{C}$  satisfies  $\beta_{\mathcal{C}} = 1$ , then  $\frac{1}{\beta_0} \leq \sum_{\mathcal{C}} f_{\mathcal{C}}$ .

1

–Mixed-level orthogonal arrays  $A_{n_1 \dots n_m}$  strength  $t$

–Mixed  $t$ -designs  $\sum_{i=1}^t \binom{n}{i} \leq \binom{n}{t}$

–Fused orthogonal array design  $\sum_{i=1}^t \binom{n}{i} \leq \binom{n}{t}$

–Split orthogonal arrays  $A_{\lambda}$   $n$   $t+T$   $n$   $n$

$\sum_{i=1}^n f_i \binom{n}{i} \leq \sum_{j=1}^N f_j \binom{N}{j}$  (D1)

**Theorem 4.3.** [6] If  $A_{\lambda}$  is split orthogonal array then

$$\sum_{i=1}^n f_i \binom{n}{i} \leq \sum_{j=1}^N f_j \binom{N}{j}$$

**Theorem 4.4.** If  $A_{\lambda}$  is split orthogonal array then

$$\sum_{i=1}^n f_i \binom{n}{i} \leq \sum_{j=1}^N f_j \binom{N}{j}$$

-wise independent

-resilient

**Theorem 4.5.** [6] The existence of  $t$ -wise independent  $t$ -resilient system is equivalent to that of split orthogonal array  $A_{\lambda}$  with  $n-t-T$ .

**Corollary 4.6** We derive the inequality

$$\sum_{i=1}^n f_i \binom{n}{i} \leq \sum_{j=1}^N f_j \binom{N}{j}$$

## References

1. J.Bierbrauer, K.Gopalakrishnan, D.R.Stinson, Orthogonal arrays, resilient functions, error correcting codes and linear programming bounds, *SIAM J.Discrete Math.* 9, 1996, 424-452.
2. J.Bierbrauer, K.Gopalakrishnan, D.R.Stinson, A note on the duality of linear programming bounds for orthogonal arrays and codes, *Bulletin of the ICA* 22, 1998, 17-24.
3. P.Delsarte, An Algebraic Approach to Association Schemes in Coding Theory, *Philips Research Reports Suppl.*, 10, 1973.
4. J.Friedman, On the bit extraction problem. *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, 1992, 314-319.
5. V.I.Levenshtein, Krawtchouk polynomials and universal bounds for codes and designs in Hamming spaces, *IEEE Trans. Inf. Theory* 41, 5, 1995, 1303-1321.
6. V.I.Levenshtein, Split orthogonal arrays and maximum independent resilient systems of functions, *Designs, Codes and Cryptography* 12, 1997, 131-160.
7. V.I.Levenshtein, Universal bounds for codes and designs, Chapter 6 in *Handbook of Coding Theory*, V.Pless and W.C.Huffman, 1998 Elsevier Science B.V., 449-648.
8. W.J. Martin, Mixed block designs, *J.Combin.Designs* 6, 2, 1998, 151-163.
9. W.J. Martin, Designs in product association schemes *Designs, Codes and Cryptography* 16, 3, 1999, 271-289.
10. S.I. Nikova, Bounds for designs in infinite polynomial metric spaces, Ph.D. Thesis, Eindhoven University of Technology, 1998.
11. S.I.Nikova, V.S.Nikov, Improvement of the Delsarte bound for  $\tau$ -designs when it is not the best bound possible, submitted in *Designs Codes and Cryptography*.
12. S.I.Nikova, V.S.Nikov, Improvement of the Delsarte bound for  $\tau$ -designs in finite polynomial metric space, to be published.
13. N.J.A. Sloane, J.Stufken, A linear programming bound for orthogonal arrays with mixed levels, *J. Stat. Plan. Inf* 56, 1996, 295-306.
14. D.R.Stinson, Resilient functions and large sets of orthogonal arrays, *Congressus Numer.* 92, 1993, 105-110.
15. F.J.MacWilliams, N.J.A.Sloane, *The Theory of Error-Correcting Codes*, North Holland, Amsterdam, 1977.

# Further Results on the Relation Between Nonlinearity and Resiliency for Boolean Functions

n l n n n

Dept. of Information Technology  
Lund University, P.O. Box 118, 221 00 Lund, Sweden  
{thomas, enes}@it.lth.se

**Abstract.** A good design of a Boolean function used in a stream cipher requires that the function satisfies certain criteria in order to resist different attacks. In this paper we study the tradeoff between two such criteria, the nonlinearity and the resiliency. The results are twofold. Firstly, we establish the maximum nonlinearity for a fixed resiliency in certain cases. Secondly, we present a simple search algorithm for finding Boolean functions with good nonlinearity and some fixed resiliency.

## 1 Introduction

l n n n n n n

n n n k w

w n w n nl n n l n l n

n n *resiliency* n n n l

n n n k w k n n l n w n

nn n ll n l l l

l l n k *nonlinearity* l n n n

n n n n n n w n w n

ll n l n n n n

n l n n n l n n n l n

l n k l kn wn

l n w l l l n n n n n

l w l n nl n l n n n

l n n w ll l n n l

n n n n nl n l n n n

n n l w kn wn l n n n

n n l n n n n l wn

n nl n n n - l n n n n n-2

l l l n n n w l

n n

n l n n n

l w k w n n n

n n n l n n n w

l n l l n n l n n w n nl n  
n l n l n l  
n n n w l n  
l l n n n nl n n n  
n ll w n n n n  
fl n kn wl n n nl n l n n  
n l n l n n w n n  
n n ll w n n w l n n  
n nl n n n l n n  
l n n n l l  
n n l n nl n l n n n n l n  
n n l n n w n l n  
n n l n

2 Preliminaries

l w w ll nl l n n n n  
n l n l n n nl n n n w  
l n l n k n  
n n l n n f x  $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$   
k n n n  
n n f x n w n n l n l

$$f(x_1, \dots, x_n) = a_0 + a_1x_1 + \dots + a_nx_n + a_{12}x_1x_2 + a_{13}x_1x_3 + \dots + a_{12} \dots x_1x_2 \dots x_n.$$

w n n l l n n  $\mathbb{F}_2$  truth table n n f x  
n n f n n  
n n n f f\_0f\_1 f\_{2^n-1}^T w f\_i  $\mathbb{F}_2$  algebraic  
degree f x n deg f n n

n n l n n n f x l ll n n  
n l n k n k n k  
n l l n k n l k l n n  
k l l n n f x l l ll  
l n n n nl n l n l ll n  
n n n n l  
n w n n n w  
l n n n  
l l n n F  $\omega$  l n n n f x n  
 $\mathbb{F}_2^n$

$$F(\omega) = \sum_x f(x) - \omega^x,$$



$$\begin{aligned} & \begin{matrix} & n & & & n & n & & n & & l & n & & n & f & x & & l & n \\ n & & & n & l & & l & n & & k & & n n & & & & n & n & l \\ & & & l & n & & k & n & & & & n & & & & & & n \\ f & x & n & & l & n & & & & & & l & k & n & & k & n \\ n & l & & & & & & & & & & & & & & & & \\ w & & & & & & n & l & & & & & & & & w & n & l \\ & & & n & & l & n & & & & & & & & & & & n \\ l & n & & k & n & w & l & l & & l & n & & l & n & & l & & k \\ & l & & & & f & x & & l & n & & n & n & & m & l & n & n \\ & & n - m & & & & l & n & & n & n & & & f & x & & & \\ n & & & n & & n & & n & & l & n & n l & n & & l & n & & l & n \\ n & n & & l & n & n & n & n & l & & n & & l & n & & n & n & & \\ w & n & l l & l & n & n l & n & & & k n & w n & n & n l & n & & n & n & l l \\ & & & & & f l & n & & & n & n l & n & & k & & & & \\ l & n & n & n & & n & N_f & n^{-1} - l_1 & w & l_1 & k & & l_1 & n & & & & \\ l l & n & & & n & & & & & & & & & & & & & \end{matrix} \end{aligned}$$

$$\begin{aligned} & \begin{pmatrix} l \\ k \end{pmatrix} \begin{pmatrix} l \\ k \end{pmatrix} \begin{pmatrix} l \\ l \end{pmatrix} n-l. \\ & l \quad f \quad x \quad n \quad deg \quad f \quad n-l_1- \\ & n \quad n \quad n \quad n \quad l \end{aligned}$$

	n					
CI	5	6	7	8	9	10
0	12	24	56	112	240	480
1	12	24	56	112	240	480
2	8	24	48	112	240	480
3	0	16	48	96	224	480
4	0	0	32	96	192	448
5	0	0	0	64	192	448
6	0	0	0	0	128	384

Table 1.  $n l n \quad N_f \quad n \quad n n$

3 Determining the Maximum Nonlinearity for Fixed Resiliency in Certain Cases

$$\begin{aligned} & \begin{matrix} n & & n & w & & l & n & & n & & l & & n & n \\ l & n & & n & n & & l & n & & n & & l & & l & n \\ n & & standard & form & & & n & n & & n & & l & & l l \end{matrix} \end{aligned}$$



	Resiliency					
$n$	0	1	2	3	4	5
5	12	12	8	0	0	0
6	26	24	24	16	0	0
7	*	*	*	*	*	0

Table 2.  $N_f$  for  $n = 5, 6, 7$  and  $l = n$

**Theorem 1.** *The maximum nonlinearity for a 1-resilient Boolean function on  $n$  variables is*

$$N_f = \begin{cases} 2^{n-1} - 2^{\frac{n-1}{2}}, & n \text{ odd} \\ 2^{n-1} - 2^{\frac{n-2}{2}}, & n \text{ even} \end{cases}$$

**Conjecture 1** *The maximum nonlinearity for a 1-resilient Boolean function  $f(x)$  on  $n$  variables is given by*

$$N_f = \begin{cases} 2^{n-1} - 2^{\frac{n}{2}}, & n \text{ even} \\ 2^{n-1} - 2^{\frac{n-1}{2}}, & n \text{ odd.} \end{cases}$$

$$N_f = 2^{n-1} - 2^{\frac{n-1}{2}}$$

**Theorem 2.** *The maximum nonlinearity for an  $(n-1)$ -resilient Boolean function  $f(x)$  on  $n$  variables is given by*

$$N_f = 2^{n-2}.$$

*Proof.* Let  $k$  be the number of variables  $x_1, \dots, x_k$  such that  $f(x) = 1$ . Then  $k \leq n-1$ . Let  $f(x)$  be a Boolean function on  $n$  variables. Then  $f(x)$  is a function of  $x_1, \dots, x_k$  and  $x_{k+1}, \dots, x_n$ . Let  $f(x)$  be a function of  $x_1, \dots, x_k$  and  $x_{k+1}, \dots, x_n$ . Let  $f(x)$  be a function of  $x_1, \dots, x_k$  and  $x_{k+1}, \dots, x_n$ .

$$\sum_{i=1}^v x_i x_{v+i} = x_{2v+1}.$$

$$N_f = 2^{n-1} - 2^{\frac{n-1}{2}}$$

4 A Novel Search Algorithm

$$\begin{aligned} & \left( \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right| \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right|^2 \end{aligned}$$

$$\begin{aligned} & F(\omega) = \sum_{x \in \mathcal{X}} \left| \frac{f(x) - f(x)}{f(x) - f(x)} \right|^2 \\ &= \sum_{x \in \mathcal{X}} \left| \frac{f(x) - f(x)}{f(x) - f(x)} \right|^2 \\ &= \sum_{x \in \mathcal{X}} \left| \frac{f(x) - f(x)}{f(x) - f(x)} \right|^2 \end{aligned}$$

$$N_f = n^{n-1} - \sum_{\omega \in \mathcal{W}} F(\omega), \quad \omega \in \mathcal{W}.$$

$$\begin{aligned} & \left( \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right| \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right|^2 \end{aligned}$$

$$\begin{aligned} & \left( \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right| \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right|^2 \end{aligned}$$

$$\begin{aligned} & \left( \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right| \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right|^2 \end{aligned}$$

$$F(Af) = Af(Af) = Af^k, \quad F(A_k f_k),$$

$$\begin{aligned} & \left( \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right| \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left| \frac{f(x_i) - f(x)}{f(x_i) - f(x)} \right|^2 \end{aligned}$$

$$\begin{array}{cccccccccccc}
& & & n & & n & w & n & & l & & n & n & f \\
l & & n & & & n & & n & F & w & & l & & F_{max} - & w & ll \\
n & & & l & n & & n & & w & ll & & n & & n & & n \\
nn & & & & n & & ll & & n & & n & & F_{max} & & l & & n \\
& & n & & F_m & & n & & n & & l & & n & & & & \\
n & & & l & & F_m & w & & & & l & & & & l & & n \\
n & n & & w & & & & n & & l & & n & n & w & & n & & n \\
& & l & & n & l & & n & w & n & & & f & x & & & \\
& & & n & & ll & & cycle & & & & l & & & n & & \\
ll & w & & & & & & & & & & & & & & & 
\end{array}$$

$$\begin{array}{cccccccccccccccc}
& & n & & n & & l & n & & l & n & n & & n & f & x \\
& & & F & n & & n & & n & F & & F_{i_1} & & F_{i_2} & & & F_{i_{2^n}} \\
n & & & n & & s & & & v & w & & v & & n & & & \\
F_{i_1}, F_{i_2}, \dots, F_{i_v} & ll & & & & w & n & & s & & & l & & n & & & \\
l & & n & & s & & n & f & x & & & & & & & & \\
& & k & & n & nl & n & & n & w & f & x & & l & & & \\
& & w & & & & & & & & & & & & & & \\
& & & & n & & n & & n & & & & & & & & 
\end{array}$$

$$\begin{array}{cccccccccccccccc}
& & & & n & & n & & n & ll & & n & & l & n & & n \\
l & & & & n & & & & n & l & & n & & & F & ll & w \\
n & & l & n & & n & & n & & & & & & & & & \\
& & f & x & & n & n & l & n & & n & & n & \mathbb{F}_2^n & & F & \omega_0 \\
\omega_0 & \mathbf{0} & n & \mathbb{F}_2^n & & n & f & x & & f & x & & x & \omega_0 & & n & & n & w & n & & n & n & \mathbb{F}_2^n \\
w & & l & n & & n & & & n & nl & n & & n & l & & & & f & x \\
& & & & n & nl & n & & n & & l & & & & n & & & n & & n \\
n & & n & f & x & & & f & x & & n & l & n & & ll & n & & n & & n \\
& & l & & n & & & w & & & & & & & & & & & 
\end{array}$$

$$F \mathbf{0} \sum_x - f(x) \sum_x - f(x) - \omega_0 x F \omega_0 .$$

$$\begin{array}{cccccccccccccccc}
& & & wn & & w & & n & & l & n & & n & & n & w & & n & w & & n & & w \\
& & n & & l & n & & n & & n & & f & x & l & W^0 & & n & & n & & n & & n \\
\mathbb{F}_2^n & & w & & & & n & & l & & & W^0 & & \omega & F & \omega & & , \omega & & \mathbb{F}_2^n \\
B & & n & n & m, m & & n & & w & & w & & l & n & l & & n & & n & & n \\
W^0 & & & l & & m & & n & & n & & lw & & l & & & & & & \\
& & & & n & & n & & l & n & & n & & & & C & & l & & n \\
n & & n & f & x & & n & & & l & n & & & & n & F & \omega & & \omega \\
n & & w & & n & ll & & & & n & & W^0 & & & n & & n & w & & \omega \\
n & & l & n & & n & & n & & & & & & & & & & & & \\
& & m & & n & n & C & & B^{-1} & & n & w & & n & & n & f & x_1, \dots, x_n & & w & & l \\
& & n & & n & w & & n & & n & & & & & & & & & & 
\end{array}$$

$$f \ x \ f \ C \ x .$$



l n w l w n  
l n n n l n l n  
n n w w l n l n  
l n m l n n n n l

## References

1. P. Camion, C. Carlet, P. Charpin and N. Sendrier, "On Correlation-Immune Functions", *Advances in Cryptology - CRYPTO'91, Lecture Notes in Computer Science*, 1233, pp. 422–433, Springer-Verlag, 1997.
2. S. Chee, S. Lee, D. Lee, S. H. Sung, "On the correlation immune functions and their nonlinearity", *Advances in Cryptology - ASIACRYPT '96, Lecture Notes in Computer Science*, 1163, pp. 232–243, Springer-Verlag, 1996.
3. L. E. Dickson (1900), *Linear Groups with an Exposition of the Galois Field Theory*, Teubner, Leipzig 1900; Dover, New York, 1958.
4. E. Filiol and C. Fontaine, "Highly Nonlinear Balanced Boolean Functions with a Good Correlation-Immunity" *Advances in Cryptology - EUROCRYPT'98, Lecture Notes in Computer Science*, 1403, pp. 475–488, Springer-Verlag, 1998.
5. R. Gallager, *Information theory and reliable communication*, 1968.
6. X. D. Hou, "On the Norm and Covering Radius of the First-Order Reed–Muller Codes". *IEEE Transactions on Information Theory*, 43(3), pp.1025–1027, 1997.
7. B. Kolman and R. E. Beck, *Elementary Linear Programming with Applications*, Academic Press, 1995.
8. S. Maitra and P. Sarkar, "Highly Nonlinear Resilient Functions Optimizing Siegenthaler's Inequality" *Advances in Cryptology - CRYPTO'99, Lecture Notes in Computer Science*, 1666, pp. 198–215, Springer-Verlag, 1999.
9. W. Meier, and O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Advances in Cryptology–EUROCRYPT'88, Lecture Notes in Computer Science*, 330, pp. 301–314, Springer-Verlag, 1988.
10. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
11. W. Millan, A. Clark and E. Dawson, "Heuristic design of cryptographically strong balanced Boolean functions" *Advances in Cryptology–EUROCRYPT'98, Lecture Notes in Computer Science*, 1403, pp. 489–499, Springer-Verlag, 1998.
12. W. Millan, A. Clark and E. Dawson, "An effective genetic algorithm for finding highly nonlinear Boolean functions", In *First International Conference on Information and Communications Security*, *Lecture Notes in Computer Science*, 1334, pp. 149–158, 1997.
13. T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only". *IEEE Trans. Comput.*, vol. C-34, pp. 81–85, 1985.
14. T. Siegenthaler, "Correlation immunity of nonlinear combining functions for cryptographic applications", *IEEE Transactions on Information Theory*, vol. IT-30, pp. 776–780, 1984.

# Combinatorial Structure of Finite Fields with Two Dimensional Modulo Metrics\*

g t z- o o<sup>1</sup> - o <sup>2</sup> g o g - <sup>3</sup>  
o g - t <sup>3</sup>  
<sup>1</sup> i i n n  
ni i i i 00 in  
edgar.martinez@ieee.org  
<sup>2</sup> ni i n i n  
ni i i i 00 in  
javi@tita.emp.uva.es  
<sup>3</sup> n i i n i  
ni i i n ni 0 00  
{mborges,mijail}@csd.uo.edu.cu

**Abstract.** i nn i n n in -  
i i n i n i ni  
nn i n n i n n  
n n i  
n i n i i n i i n  
n i i n i  
i i i n n i n i  
n i n i -  
i i n i n i  
i i

## 1 Introduction

t to fly o t o t t g  
t - o t g t o o o o t o  
t o t o t t to o t g  
o g t o y o t t t y o to o t  
t o to g o o t o t to o t  
o o t t

### 1.1 Gaussian and Einsestein-Jacobi Numbers

t o g z t o o t g y g -  
algebraic integer t oot o o o y o o  

---

\* i n n n i n n -  
i n i n i  
i -0 1

**Case A: Gaussian Integers**

constructing  $\mathbb{F}_{p^2}$  with the irreducible polynomial  $x^2 - \alpha$ .

in i ini i

### Case B: Einsestein-Jacobi Integers

Case B: Einsestein-Jacobi Integers

### 1.3 Metrics over $\mathbb{Z}[\delta]/\pi\mathbb{Z}[\delta]$

o o o ot t o t t o  
t g ot t t t o t o g  
y - 3 o - o go

## Mannheim Metric

**Mannheim Metric**

$$g_{\alpha\beta} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\sum_{j=0}^{n-1} M_j \mathbf{x}$$

$$\mathbf{t} = \pi^n \mathbf{g} \quad \mathbf{y} = \mathbf{M} \mathbf{x} \quad \mathbf{y} = \mathbf{M} \mathbf{x} - \mathbf{y}$$

## Hexagonal Metric

Hexagonal Metric				to				hexagonal weight			
o	t	-	o	$\gamma$	t		t	o	t	t	t
—	—	—	—	o	0 to $\gamma$	t	o	to	o	$z^m$	o
o	$2m$	o	—	o		o	o		t	g t	t
t	to			$\pi$	$n$						to

## 2 Association Schemes

## 2.1 Transitive Actions

t o o g o o o t o o t t o g o  
t g o t t o o t t o o g t t o  
g o G o t o t t o t o t o t -  
G - - - G - o o t t t o o t o t  
y - G - t t o t o t t o o t t t z



**Definition 1** An association scheme with  $d$  classes is a pair  $(X, \{R_i\}_{i=0}^d)$  of a finite set  $X$  and a set of relations  $R_i$  on  $X$  satisfying the following rules:

1.  $R_0 = \{(x, x) \mid x \in X\}$  (the diagonal relation)
2.  $\{R_i\}_{i=0}^d$  is a partition on  $X \times X$ .
3.  $|R_i| = |R_j|$  s.t.  $t_i = t_j$ , where  $t_i = \sum_{x \in X} |R_i(x, \cdot)|$
4. For each election of  $x \in X$ , the number:  $k_{ij} = |\{y \in X \mid (x, y) \in R_j\}|$  is constant for all  $x \in X$

Let  $A_i$  be the  $|X| \times |X|$  matrix with entries  $A_i(x, y) = 1$  if  $(x, y) \in R_i$  and 0 otherwise. Then the following conditions are equivalent to the definition:

- 1'.  $A_0$  (identity matrix)
- 2'.  $\sum_{k=0}^d A_k = J$ , where  $J$  is the matrix where all entries are 1.
- 3'.  $A_i A_j = A_j A_i$  such that  $t_i = t_j$
- 4'.  $k_{ij} = \sum_{k=0}^d A_{ij}^k$

Let  $\mathcal{B}$  be the  $|X| \times |X|$  matrix with entries  $\mathcal{B}(x, y) = \frac{1}{|X|} \sum_{i=0}^d A_i(x, y) t_i$ . Then  $\mathcal{B}$  is a symmetric matrix and  $\mathcal{B}^2 = \mathcal{B}$ . The eigenvalues of  $\mathcal{B}$  are  $1, \frac{t_1}{|X|}, \dots, \frac{t_{d-1}}{|X|}$ . The eigenvectors of  $\mathcal{B}$  are the vectors  $(1, 1, \dots, 1)$  and the vectors  $(\chi_1(x), \chi_1(y), \dots, \chi_1(z))$  where  $\chi_1$  is a non-trivial character of the group  $G$ .

### 2.3 Constructing the Mannheim Scheme

Let  $G$  be a finite group and  $\chi$  a non-trivial character of  $G$ . Let  $X = G$  and  $R_i = \{(x, y) \in G \times G \mid \chi(yx^{-1}) = \omega^i\}$  where  $\omega$  is a primitive  $n$ -th root of unity. Then  $(X, \{R_i\}_{i=0}^{n-1})$  is an association scheme with  $n$  classes. The matrices  $A_i$  and  $\mathcal{B}$  are given by  $A_i(x, y) = 1$  if  $\chi(yx^{-1}) = \omega^i$  and 0 otherwise, and  $\mathcal{B}(x, y) = \frac{1}{|G|} \sum_{i=0}^{n-1} A_i(x, y) t_i$  where  $t_i = \sum_{x \in G} |R_i(x, \cdot)|$ .

[illegible]
$$0-0 \quad -0- \quad 0-1 \quad - \quad - \quad 0-2 \quad - \quad - \quad 0-3 \quad - \quad 0 \quad 3-$$
$$\begin{array}{cccccccccccc} 0 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & & 0 & & 0 & 0 & 0 & & 0 & 0 & & 0 & 0 & 0 & 0 & & 0 & 0 & 0 \\ 2 & & 0 & 0 & 0 & 0 & & 0 & & 0 & & 0 & 0 & 0 & & 0 & & 0 & 0 \\ 3 & & 0 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & & 0 & & 0 & 0 \end{array}$$

Note that each matrix  $\mathbf{A}_i$  is represented only by its first row since they are circulant, and hence their eigenvalues are calculated easily ( see [3]), and they are:





and

$$\frac{\sum_{j_1+2j_2+\dots+nj_n=n} \prod_{k=1}^n \Gamma_k^{j_k}}{\Gamma_0 \Gamma_1 \Gamma_2 \Gamma_3^{j_1} \Gamma_0^2 \Gamma_1^2 \Gamma_2^2 \Gamma_3^{j_1}} - F -$$

respectively.

**Example 4** If we let  $\Gamma = \Gamma_i$  if  $\Gamma = \Gamma_i$  we get  $\Gamma_k = \sum_{i=1}^k \Gamma_i$ , the resulting pattern sum is an homogeneous polynomial in  $\Gamma_1, \dots, \Gamma_l$ . For example, consider  $\Gamma = \Gamma_3$  represented as  $\Gamma = \Gamma_{3+2i}$  (see example 1), and consider the case  $\Gamma = \Gamma_3$ . The pattern sum is:

$$\Gamma = \frac{\sum_{j_1+2j_2=2} \prod_{k=1}^2 \Gamma_k^{j_k}}{\Gamma_0 \Gamma_1 \Gamma_2 \Gamma_3^{j_1} \Gamma_0^2 \Gamma_1^2 \Gamma_2^2 \Gamma_3^{j_1}} - \Gamma_0 \Gamma_1 \Gamma_2 \Gamma_3^2 - \Gamma_0^2 \Gamma_1^2 \Gamma_2^2 \Gamma_3^2$$

Indeed, we get no much information since we get all degree two monomials as the possible patterns of errors, i.e. all the combinations of two errors taken from the orbits. Following example gives us some more information:

**Example 5** If we let  $\Gamma = \Gamma_{d(i)}$  if  $\Gamma = \Gamma_i$ . For example, consider  $\Gamma = \Gamma_3$  represented as  $\Gamma = \Gamma_{3+2i}$  (see example 1), and consider the case  $\Gamma = \Gamma_3$ . The pattern sum in this case is:

$$\Gamma = \Gamma_0 \Gamma_1 \Gamma_2^2 - \Gamma_0^2 \Gamma_1^2 \Gamma_2^2$$

For example, an error has weight  $\Gamma_0 \Gamma_2$  if it is an error pattern of distance two, thus the number of such patterns is the coefficient of the monomial  $\Gamma_0 \Gamma_2$  in  $\Gamma$

$$\Gamma = \Gamma^{d(i)} \Gamma - \Gamma_i \Gamma_0 \Gamma_1 \Gamma_2^2 - \Gamma_0^2 \Gamma_1^2 \Gamma_2^2$$

## 4 Conclusions

$$\Gamma = \Gamma_0 \Gamma_1 \Gamma_2^2 - \Gamma_0^2 \Gamma_1^2 \Gamma_2^2$$

t o t t o o t o  
 o t t o o t g t t t o t g  
 o t t o y o y o o  
 t o o o t t y o t o y o o  
 t o o g o o o g t o t t o o o o t  
 o t t o t t o o t o t o o t  
 t t t o o t g t t o y y t  
 t o t t o t o o t o o t o o  
 t g o —  $I-I$  —  $I-II$  —  $I-o$  t t 0  
 o t g o t  
 t t g t o o t t o t t o o t o o  
 t t t o t o t t o t t o t g  
 o t o y o y o <sup>2</sup> o t o t y o t o  
 o o o t g t o

## References

- 1 J.L. Alperin , R.B. Bell in -  
i l in 1  
E. Bannai, T. Ito n in  
i i 1  
N. Biggs n i i n i ni i  
1  
B. Bollobàs in i in  
1  
P. Camion in n in 1 1  
1 n - n l  
Hardy,G.H., Wright,E.M.  
in i in i i n l  
K. Huber n n n 0 1  
0 - 1 1  
K. Huber  
1 1- 1  
P. Delsarte ni ii 1  
10 P. Delsarte,V.I. Levenshtein n n n 0 1  
11 W. Ledermann i ni i  
1  
1 H. Lehmann "  
" i ni i " i n l  
1 E. Martínez, F.J. Galán *Combinatorial structure of arithmetic codes.* in  
n in n n i n 1 n

- 1 **E. Martínez,M.A. Borges, M. Borges** *Combinatorial structure of rings of complex integers with Mannheim metric.* 1 n in i -  
in n n
- 1 **E. Martínez** *Computations on character tables of association schemes.* -  
in i n i in 0 in - 1
- 1 **H. Tarnanen** ni n  
n i n 1
- 1 **P.Solé** i  
i 1 1 1 1
- 1 **P.Solé** -  
in i 10 1 1 1

M. Walker (Ed.): IMA - Crypto & Coding'99, LNCS 1746, pp. 56–62, 1999.  
© Springer-Verlag Berlin Heidelberg 1999



### 3 Alternative Constructs

## 4 Correlation Properties

[illegible]

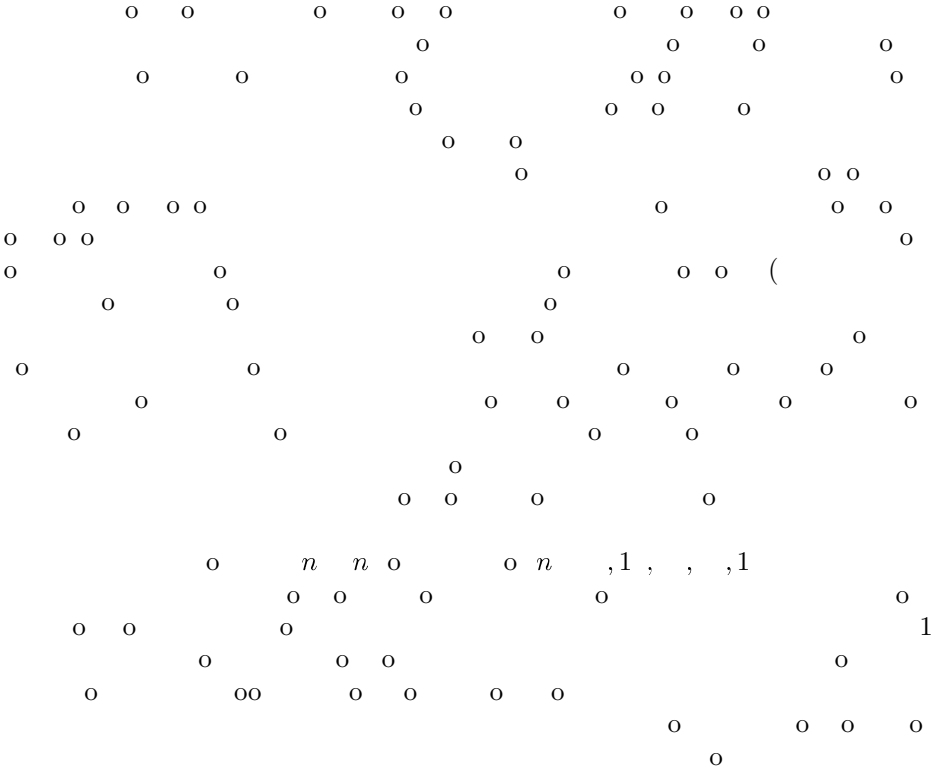
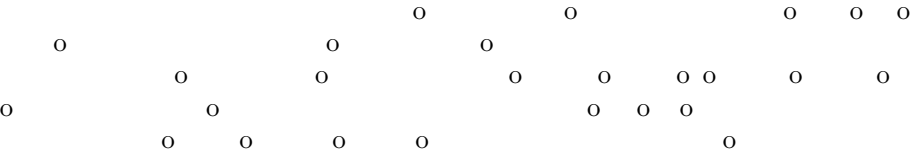


Table 1.

Set Size	Number of Sets	Peak Correlation Value
8× 8	7	4
16× 16	15	8
32× 32	31	8
64× 64	63	16
128× 128	127	20
256× 256	256	32

5 Mean Square Correlation Parameters





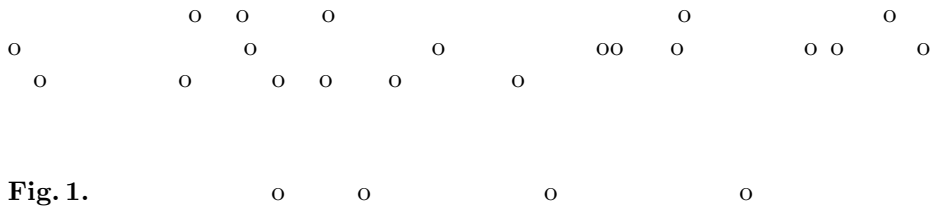
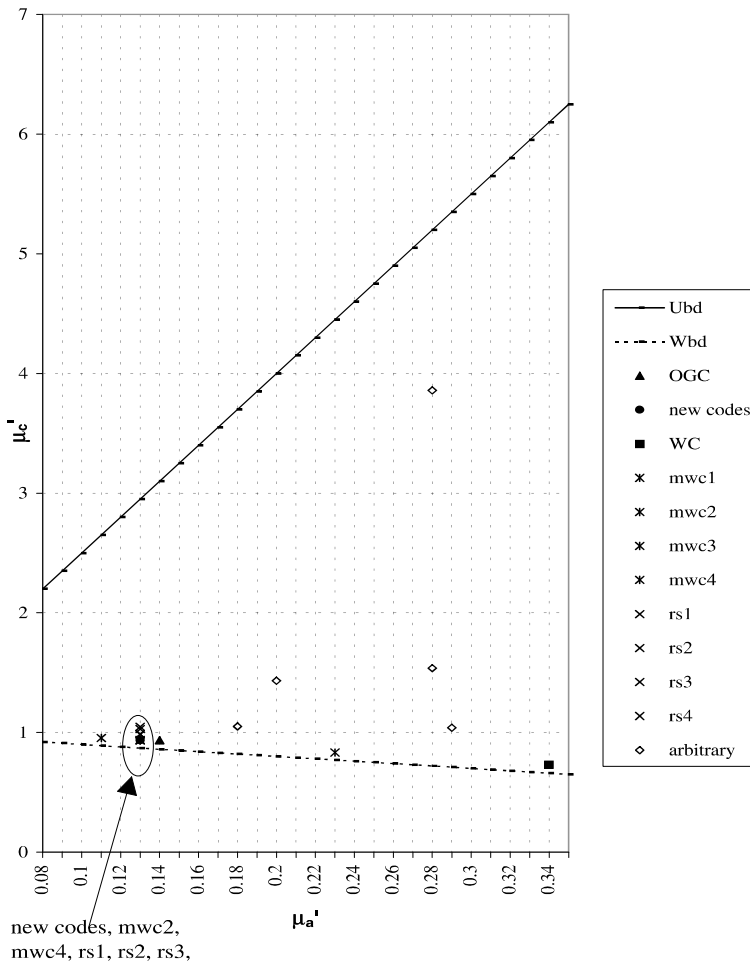


Fig. 1.  
1 1





# New Self-Dual Codes over $GF(5)$

Stelios Georgiou and Christos Koukouvinos

Department of Mathematics,  
National Technical University of Athens,  
Zografou 15773, Athens, Greece.

**Abstract.** Self-dual codes and orthogonal designs have been studied for a long time as separate research areas. In the present paper we show a strong relationship between them. The structure of orthogonal designs is such as to allow us a much faster and more systematic search for self-dual codes over  $GF(5)$ .

Using our method we constructed the following linear self-dual codes over  $GF(5)$ : (i)  $[4,2,2]$ , (ii)  $[8,4,4]$ , (iii)  $[12,6,6]$ , (iv)  $[16,8,6]$ , (v)  $[20,10,8]$ , (vi)  $[24,12,9]$ , (vii)  $[28,14,10]$ . The codes (i), (ii), (iii), (v) are extremal. A  $[28,14,10]$  code is constructed here for the first time.

*Key words and phrases:* Self-dual codes, construction, orthogonal designs.

## 1 Introduction

We first give some basic definitions which are needed in order to explain our method for the construction of self-dual codes. Self-dual codes are important because many of the best codes known are of this type and have a rich mathematical theory. An *orthogonal design* of order  $n$  and type  $(s_1, s_2, \dots, s_u)$  ( $s_i > 0$ ), denoted  $OD(n; s_1, s_2, \dots, s_u)$ , on the commuting variables  $x_1, x_2, \dots, x_u$  is an  $n \times n$  matrix  $D$  with entries from the set  $\{0, \pm x_1, \pm x_2, \dots, \pm x_u\}$  such that

$$DD^T = \left( \sum_{i=1}^u s_i x_i^2 \right) I_n$$

Alternatively, the rows of  $D$  are formally orthogonal and each row has precisely  $s_i$  entries of the type  $\pm x_i$ . In [2], where this was first defined, it was mentioned that

$$D^T D = \left( \sum_{i=1}^u s_i x_i^2 \right) I_n$$

and so our alternative description of  $D$  applies equally well to the columns of  $D$ . It was also shown in [2] that  $u \leq \rho(n)$ , where  $\rho(n)$  (Radon's function) is defined by  $\rho(n) = 8c + 2^d$ , when  $n = 2^a b$ ,  $b$  odd,  $a = 4c + d$ ,  $0 \leq d < 4$ . For more details and construction methods of orthogonal design see [3].

In this paper we restrict our attention in two variable orthogonal designs, i.e. in the case where  $u = 2$ .

For our consideration we also need some facts from coding theory. Our terminology and notation follow [6]. Let  $F = GF(q)$  be the field with  $q$  elements where  $q$  is a prime power. An  $[n, k]$  linear code  $C$  over  $F$  is a  $k$ -dimensional vector subspace of  $F^n$ . In particular, codes over  $GF(2)$  and  $GF(3)$  are said binary and ternary codes, respectively. The elements of  $C$  are called codewords and the weight of the codeword is the number of its non-zero coordinates. A minimum weight is the smallest weight among non-zero codewords. An  $[n, k]$  code with a minimum weight  $d$  is called an  $[n, k, d]$  code. Two binary codes are equivalent if one can be obtained from the other by a permutation of the coordinates.

The dual code  $C^\perp$  of  $C$  is defined as  $C^\perp = \{x \in F^n \mid x \cdot y = 0 \text{ for all } y \in C\}$ . If  $C = C^\perp$ ,  $C$  is called a self-orthogonal code.  $C$  is called self-dual if  $C = C^\perp$ . Furthermore  $C$  is called doubly-even if the weights of all codewords of  $C$  are a multiple of four. A self-dual code is called singly-even if there exists at least one codeword whose weight is  $2 \pmod{4}$ .

A self-dual code  $C$  is called extremal if  $C$  has the largest possible minimum weight. The known bounds of  $d$  for  $q = 2, 3, 4$  are given in [7] and [8]. In particular the following theorem is known.

**Theorem 1** ([8]) *The minimum distance  $d$  of a self-dual  $[n, n/2]$  code  $C$  satisfies*

$$d \leq \begin{cases} 2 \left\lfloor \frac{n}{8} \right\rfloor + 2 & \text{if } q = 2 \text{ and } C \text{ is singly-even} \\ 4 \left\lfloor \frac{n}{24} \right\rfloor + 4 & \text{if } q = 2 \text{ and } C \text{ is doubly-even} \\ 3 \left\lfloor \frac{n}{12} \right\rfloor + 3 & \text{if } q = 3 \\ 2 \left\lfloor \frac{n}{6} \right\rfloor + 2 & \text{if } q = 4 \text{ and } C \text{ is even.} \end{cases}$$

For each length, the details of the largest possible minimum weight is listed in Table I in [1]. Conway and Sloane [1] also gave a list of the possible weight enumerators of binary extremal self-dual codes. The existence of some extremal self-dual codes is an open question in [1].

## 2 The Method

In this section we will show how we can use an orthogonal design in order to obtain a linear self-dual code over  $GF(5)$ .

We consider an orthogonal design  $OD(n; s_1, s_2)$ . Then we replace the first variable by 1 and the second variable by 2. This replacement of course does not affect the orthogonality of the rows, and let us denote the derived matrix by  $A$ . We shall take the elements of  $GF(5)$  to be either  $\{0, 1, 2, 3, 4\}$  or  $\{0, -1, -2\}$  using whichever form is more convenient.

On the other hand since there are more orthogonal matrices with elements from  $GF(5)$  than orthogonal designs with elements from a set of commuting variables, we use both of them in order to construct the desired codes.

**Lemma 1** *If we say  $c = \sum_{i=1}^u s_i x_i^2$ , where in our case  $u = 2$ , then the matrix  $C = [aI_n, A]$  is the generator matrix of a  $[2n, n, d; 5]$  linear self-dual code if and only if  $c + a^2$  is divisible by 5.*

*Proof.* We have that

$$CC^T = [aI_n, A][aI_n, A]^T = (c + a^2)I_n.$$

Thus if  $c + a^2$  is divisible by 5 then  $CC^T = 0_n$  over  $GF(5)$ , where  $0_n$  is the  $n - n$  matrix whose entries are all zero, and then the matrix  $C = [aI_n, A]$  is the generator matrix of a  $[2n, n, d; 5]$  linear self-dual code. On the other hand if the matrix  $C = [aI_n, A]$  is the generator matrix of a  $[2n, n, d; 5]$  linear self-dual code then  $CC^T = 0_n$  over  $GF(5)$  and then  $c + a^2$  is divisible by 5. —

**Example 1** *We consider the following orthogonal design  $OD(8; 2, 6)$ .*

$$D = \begin{bmatrix} b & a & a-b & b & b-b & b \\ a & b-b & a & b & b & b-b \\ -a & b & b & a-b & b-b-b \\ b-a & a & b & b-b & -b-b \\ -b-b & b-b & b & a & a-b \\ -b-b-b & b & a & b-b & a \\ b-b & b & b-a & b & b & a \\ -b & b & b & b & b-a & a & b \end{bmatrix}.$$

*Then we replace the first variable by 1 and the second variable by 2. This replacement of course does not affect the orthogonality of the rows, and let us denote the derived matrix by  $A$ . We shall take the elements of  $GF(5)$  to be  $\{0, 1, 2, 3, 4\}$ . Then  $[2I_8, A]$  is the generator matrix of a  $[16, 8, 6; 5]$  linear self-dual code where  $A$  is the following matrix.*

$$A = \begin{bmatrix} 2 & 1 & 1 & 3 & 2 & 2 & 3 & 2 \\ 1 & 2 & 3 & 1 & 2 & 2 & 2 & 3 \\ 4 & 2 & 2 & 1 & 3 & 2 & 3 & 3 \\ 2 & 4 & 1 & 2 & 2 & 3 & 3 & 3 \\ 3 & 3 & 2 & 3 & 2 & 1 & 1 & 3 \\ 3 & 3 & 3 & 2 & 1 & 2 & 3 & 1 \\ 2 & 3 & 2 & 2 & 4 & 2 & 2 & 1 \\ 3 & 2 & 2 & 2 & 2 & 4 & 1 & 2 \end{bmatrix}.$$

*Its weight enumerator is*

$$\begin{aligned} W(z) = & 1 + 160z^6 + 192z^7 + 2880z^8 + 5568z^9 + 26848z^{10} + \\ & + 37824z^{11} + 89568z^{12} + 84480z^{13} + 91392z^{14} + \\ & + 39936z^{15} + 11776z^{16}. \end{aligned}$$

It is obvious that any orthogonal design with two variables can give a linear code over  $GF(5)$  and if there exist  $a - GF(5)$  such that Lemma 1 holds then this code is self-dual, but in order to find a large enough minimum weight  $d$  we must try a lot of orthogonal designs and orthogonal matrices. From the description of our method it is clear that this can also be applied in the construction of self-dual codes over  $GF(2)$  and  $GF(3)$ . Thus we are able to construct a series of the previously known linear self-dual codes over  $GF(2)$  and  $GF(3)$  by this method.

**Example 2** *We consider the following orthogonal design  $OD(4; 2, 2)$ .*

$$D = \begin{bmatrix} a & b & a-b \\ b & a-b & a \\ -a & b & a & b \\ b-a & b & a \end{bmatrix}.$$

*Then we replace both variables by 1. This replacement of course does not affect the orthogonality of the rows, and let us denote the derived matrix by  $A$ . We shall take the elements of  $GF(3)$  to be  $\{0, 1, 2\}$ . Then  $[I_4, A]$  is the generator matrix of a  $[8, 4, 4; 3]$  linear extremal self-dual code where  $A$  is the following matrix.*

$$A = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{bmatrix}.$$

*Its weight enumerator is*

$$W(z) = 1 + 24z^4 + 16z^5 + 32z^6 + 8z^8.$$

### 3 The Results

In this section we present the results that we find using either orthogonal design or orthogonal matrices. In particular we construct the following linear self-dual codes over  $GF(5)$ : 1.  $[4, 2, 2]$ , 2.  $[8, 4, 4]$ , 3.  $[12, 6, 6]$ , 4.  $[16, 8, 6]$ , 5.  $[20, 10, 8]$ , 6.  $[24, 12, 9]$ , 7.  $[28, 14, 10]$ . The codes (1), (2), (3), (5) are extremal. Self-dual codes over  $GF(5)$  with same parameters were constructed, but with a different method, in [4] and [5]. A  $[24, 12, 9]$  code was also constructed in [4]. A  $[28, 14, 10]$  code is constructed here for the first time. Although it has not been proved yet if this code is extremal or not its minimum distance is quite large.

1. The matrix  $[I_2, A]$  is the generator matrix of an  $[4, 2, 2]$  extremal singly-even self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}.$$

Its weight enumerator is

$$W(z) = 1 + 8z^2 + 16z^4.$$

2. The matrix  $[2I_4, A]$  is the generator matrix of an  $[8, 4, 4]$  extremal self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 3 & 3 & 2 & 3 \\ 3 & 3 & 3 & 2 \\ 3 & 2 & 3 & 3 \\ 2 & 3 & 3 & 3 \end{bmatrix}.$$

Its weight enumerator is

$$W(z) = 1 + 48z^4 + 32z^5 + 288z^6 + 128z^7 + 128z^8.$$

3. The matrix  $[I_6, A]$  is the generator matrix of an  $[12, 6, 6]$  extremal self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 3 & 3 & 4 & 3 & 4 & 0 \\ 4 & 3 & 3 & 0 & 3 & 4 \\ 3 & 4 & 3 & 4 & 0 & 3 \\ 2 & 0 & 1 & 3 & 4 & 3 \\ 1 & 2 & 0 & 3 & 3 & 4 \\ 0 & 1 & 2 & 4 & 3 & 3 \end{bmatrix}.$$

Its weight enumerator is

$$W(z) = 1 + 440z^6 + 528z^7 + 2640z^8 + 2640z^9 + 5544z^{10} + 2640z^{11} + 1192z^{12}.$$

4. The matrix  $[2I_8, A]$  is the generator matrix of a  $[16, 8, 6]$  self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 2 & 1 & 1 & 3 & 2 & 2 & 3 & 2 \\ 1 & 2 & 3 & 1 & 2 & 2 & 2 & 3 \\ 4 & 2 & 2 & 1 & 3 & 2 & 3 & 3 \\ 2 & 4 & 1 & 2 & 2 & 3 & 3 & 3 \\ 3 & 3 & 2 & 3 & 2 & 1 & 1 & 3 \\ 3 & 3 & 3 & 2 & 1 & 2 & 3 & 1 \\ 2 & 3 & 2 & 2 & 4 & 2 & 2 & 1 \\ 3 & 2 & 2 & 2 & 2 & 4 & 1 & 2 \end{bmatrix}.$$

Its weight enumerator is

$$\begin{aligned} W(z) = & 1 + 160z^6 + 192z^7 + 2880z^8 + 5568z^9 + 26848z^{10} + \\ & + 37824z^{11} + 89568z^{12} + 84480z^{13} + 91392z^{14} + \\ & + 39936z^{15} + 11776z^{16}. \end{aligned}$$

5. The matrix  $[I_{10}, A]$  is the generator matrix of an  $[20, 10, 8]$  extremal self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 3 & 3 & 4 & 1 & 1 & 3 & 0 & 0 & 0 & 2 \\ 1 & 3 & 3 & 4 & 1 & 2 & 3 & 0 & 0 & 0 \\ 1 & 1 & 3 & 3 & 4 & 0 & 2 & 3 & 0 & 0 \\ 4 & 1 & 1 & 3 & 3 & 0 & 0 & 2 & 3 & 0 \\ 3 & 4 & 1 & 1 & 3 & 0 & 0 & 0 & 2 & 3 \\ 2 & 3 & 0 & 0 & 0 & 3 & 1 & 1 & 4 & 3 \\ 0 & 2 & 3 & 0 & 0 & 3 & 3 & 1 & 1 & 4 \\ 0 & 0 & 2 & 3 & 0 & 4 & 3 & 3 & 1 & 1 \\ 0 & 0 & 0 & 2 & 3 & 1 & 4 & 3 & 3 & 1 \\ 3 & 0 & 0 & 0 & 2 & 1 & 1 & 4 & 3 & 3 \end{bmatrix}.$$

Its weight enumerator is

$$\begin{aligned} W(z) = & 1 + 2280z^8 + 23408z^{10} + 72960z^{11} + 241680z^{12} + 437760z^{13} + \\ & + 1203840z^{14} + 1586880z^{15} + 2229840z^{16} + 1901520z^{17} + \\ & + 1418160z^{18} + 528960z^{19} + 118336z^{20}. \end{aligned}$$

6. The matrix  $[2I_{12}, A]$  is the generator matrix of a  $[24, 12, 9]$  self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 0 & 1 & 4 \\ 1 & 4 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 4 & 0 \\ 1 & 1 & 4 & 1 & 1 & 1 & 4 & 1 & 1 & 4 & 0 & 1 \\ 4 & 4 & 4 & 4 & 1 & 1 & 1 & 0 & 4 & 4 & 4 & 1 \\ 4 & 4 & 4 & 1 & 4 & 1 & 0 & 4 & 1 & 4 & 1 & 4 \\ 4 & 4 & 4 & 1 & 1 & 4 & 4 & 1 & 0 & 1 & 4 & 4 \\ 4 & 4 & 1 & 4 & 0 & 1 & 4 & 1 & 1 & 1 & 1 & 1 \\ 4 & 1 & 4 & 0 & 1 & 4 & 1 & 4 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 1 & 4 & 0 & 1 & 1 & 4 & 1 & 1 & 1 \\ 0 & 4 & 1 & 1 & 1 & 4 & 4 & 4 & 4 & 4 & 1 & 1 \\ 4 & 1 & 0 & 1 & 4 & 1 & 4 & 4 & 4 & 1 & 4 & 1 \\ 1 & 0 & 4 & 4 & 1 & 1 & 4 & 4 & 4 & 1 & 1 & 4 \end{bmatrix}.$$

Its weight enumerator is

$$\begin{aligned} W(z) = & 1 + 1056z^9 + 11088z^{10} + 36960z^{11} + 212352z^{12} + 591360z^{13} + \\ & + 2382336z^{14} + 5287040z^{15} + 13796640z^{16} + 23037696z^{17} + \\ & + 39528720z^{18} + 46163040z^{19} + 49252896z^{20} + 35604800z^{21} + \\ & + 20240352z^{22} + 6832320z^{23} + 1161968z^{24}. \end{aligned}$$

7. The matrix  $[2I_{14}, A]$  is the generator matrix of a  $[28, 14, 10]$  self-dual code where  $A$  is the following matrix.

$$A = \begin{bmatrix} 3 & 3 & 3 & 0 & 3 & 1 & 3 & 3 & 3 & 4 & 3 & 1 & 1 & 0 \\ 3 & 3 & 3 & 3 & 0 & 3 & 1 & 0 & 3 & 3 & 4 & 3 & 1 & 1 \\ 1 & 3 & 3 & 3 & 3 & 0 & 3 & 1 & 0 & 3 & 3 & 4 & 3 & 1 \\ 3 & 1 & 3 & 3 & 3 & 3 & 0 & 1 & 1 & 0 & 3 & 3 & 4 & 3 \\ 0 & 3 & 1 & 3 & 3 & 3 & 3 & 3 & 1 & 1 & 0 & 3 & 3 & 4 \\ 3 & 0 & 3 & 1 & 3 & 3 & 3 & 4 & 3 & 1 & 1 & 0 & 3 & 3 \\ 3 & 3 & 0 & 3 & 1 & 3 & 3 & 3 & 4 & 3 & 1 & 1 & 0 & 3 \\ 2 & 0 & 4 & 4 & 2 & 1 & 2 & 3 & 3 & 1 & 3 & 0 & 3 & 3 \\ 2 & 2 & 0 & 4 & 4 & 2 & 1 & 3 & 3 & 3 & 1 & 3 & 0 & 3 \\ 1 & 2 & 2 & 0 & 4 & 4 & 2 & 3 & 3 & 3 & 3 & 1 & 3 & 0 \\ 2 & 1 & 2 & 2 & 0 & 4 & 4 & 0 & 3 & 3 & 3 & 3 & 1 & 3 \\ 4 & 2 & 1 & 2 & 2 & 0 & 4 & 3 & 0 & 3 & 3 & 3 & 3 & 1 \\ 4 & 4 & 2 & 1 & 2 & 2 & 0 & 1 & 3 & 0 & 3 & 3 & 3 & 3 \\ 0 & 4 & 4 & 2 & 1 & 2 & 2 & 3 & 1 & 3 & 0 & 3 & 3 & 3 \end{bmatrix}.$$

Its weight enumerator is

$$\begin{aligned} W(z) = & 1 + 3500z^{10} + 9856z^{11} + 99820z^{12} + 362152z^{13} + 1938224z^{14} + \\ & + 6041504z^{15} + 22861496z^{16} + 57103424z^{17} + 154245868z^{18} + \\ & + 300198752z^{19} + 575888012z^{20} + 833084840z^{21} + 1106507192z^{22} + \\ & + 1116111808z^{23} + 955594024z^{24} + 598184608z^{25} + 281403808z^{26} + \\ & + 81992064z^{27} + 11884672z^{28}. \end{aligned}$$

## References

1. J.H.Conway and N.J.A.Sloane, A new upper bound on the minimal distance of self-dual codes, *IEEE Trans. Inform. Theory*, 36 (1990), 1319-1333.
2. A.V.Geramita, J.M.Geramita, and J.Seberry Wallis, Orthogonal designs, *Linear and Multilinear Algebra*, 3 (1976), 281-306.
3. A.V.Geramita, and J.Seberry, *Orthogonal designs: Quadratic forms and Hadamard matrices*, Marcel Dekker, New York-Basel, 1979.
4. M.Harada, Double circulant self-dual codes over GF(5), *Ars Combinatoria*, to appear.
5. J.S.Leon, V.Pless, and N.J.A.Sloane, Self-dual codes over GF(5), *J. Combin. Theory Ser. A*, 32 (1982), 178-194.
6. F.J.MacWilliams and N.J.A.Sloane, *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977.
7. E.Rains and N.J.A. Sloane, Self-dual codes, in *Handbook of Coding Theory*, eds. V.Pless et al., Elsevier, Amsterdam, 1998.
8. V.D.Tonchev, Codes, in *The CRC Handbook of Combinatorial Designs*, ed. C.J.Colbourn and J.H.Dinitz, CRC Press, Boca Raton, Fla., 1996, 517-543.



$l$   $l$   
 $m$   $x$   $m$   
 $x$   $m$   $.$   
 $m$   
 $x$   $m$   $m$   
 $m$   $m$   $m$   $.$   $m$   
 $m$   $.$  *Algebraic*  
*Normal Form*  $.$   
 $m$   $.$   
 $1$   $x$   $m$   $.$   
 $m$   $m$   $m$   $.$   
 $m$   $m$   
 $k$   $m$   $j$   $x$   
 $m$   $m$   
 $m$   $j$   
 $n$   $m$   $.$   
 $x$   $.$   
 $m$   $.$

## 2 Preliminaries

$\mathbb{F}_2$   $m$   
 $wt\ 1\ f\ n - wt\ f.$   $m$

### 2.1 Boolean Functions

$x$   $m$   
 $f$   $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2.$   
 $f(x_1, x_2, \dots, x_n) = \sum_{\alpha \in \mathbb{F}_2^n} a_{\alpha} x^{\alpha} \quad a_{\alpha} \in \mathbb{F}_2$   
 $x^{\alpha} = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \quad \alpha_i \in \mathbb{F}_2.$   
 $a_{\alpha}$   $m-1$   
 $a_{\alpha} = g(\alpha) = f(\beta_1, \beta_2, \dots, \beta_n)$   
 $\beta = \alpha$   
 $\beta - \alpha$   $\mathbb{F}_2^n.$   
 $\beta - \alpha = \beta_i - \alpha_i = -i.$   $m$



## 2.2 Designs and Intersecting Families

**Definition 1** A  $t-(v, k, \lambda)$  design is a pair  $(\mathcal{P}, \mathcal{B})$  where  $\mathcal{P}$  is a  $v$ -element set of points and  $\mathcal{B}$  is a collection of  $k$ -element subsets of  $\mathcal{P}$  (blocks) with the property that every  $t$ -element subset of  $\mathcal{P}$  is contained in exactly  $\lambda$  blocks.

*Balanced Incomplete Block Design*

$t = 2$   $v = v$   $k = k$   $r = r$   $b = b$   $\lambda = \lambda$   $j = j$   $x = x$

admissible parameters

- $r(k-1) = \lambda(v-1)$
- $bk = vr$
- $v > b$

$k \in \mathbb{F}_2$   $m = m$   $m = m$

$b = v$   $\binom{v}{k} = \binom{v}{k}$   $k = k$   $r = r$   $\lambda = \lambda$   $m = m$  *symmetric*

$k = k$   $s = s$   $m = m$   $k = k$   $s = s$

$m = m$   $v = v$   $n = n$   $n = n$   $m = m$   $k = k$

$j = j$   $m = m$   $m = m$

**Definition 2** [4] Let  $\mathcal{F}$  be a family of subsets of a  $n$ -set.  $\mathcal{F}$  is said intersecting if

$$|A \cap B| \geq k, \quad |A \cap B| \geq k, \quad |A \cap B| \geq k$$

$k = k$   $m = m$   $j = j$   $m = m$

**Proposition 2** Let  $\mathcal{F}$  be an intersecting family of subsets of a  $n$ -set. Then

$$|\mathcal{F}| \leq \binom{n-1}{k}.$$

There exist intersecting families reaching this upper bound.

$m = m$   $n-1 = n-1$   $m = m$   $m = m$

$m = m$   $x = x$   $m = m$   $m = m$   $j = j$

$1 = 1$   $1 = 1$   $1 = 1$

### 3 ANF and SBIBD

$$\begin{aligned}
 & \text{mm} \quad \text{m k} \quad \text{m} \\
 & \quad \quad \quad \text{j} \\
 & \quad \quad \quad \cdot \quad \text{m} \\
 & \text{m} \quad \text{m} \quad \text{j} \quad \text{m} \\
 & \quad \quad \quad t - v, k, \lambda \\
 & \quad \quad \quad \text{k} \quad \text{m} \\
 & \text{m} \quad \quad \quad \cdot
 \end{aligned}$$

**Definition 3** Let  $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  a Boolean function. Let  $\alpha = \text{supp } f$  and  $\beta = \mathbb{N}_n$ . Then  $f$  and the structure  $(\alpha, \beta)$  will be said associate. When  $(\alpha, \beta)$  is a  $t - n, k, \lambda$  design for some parameters  $n, k$  and  $\lambda$ , it is called the associate design of  $f$ .

$$\begin{aligned}
 & \quad \quad \quad \text{---}, \text{---} \quad \quad \quad \text{k} \quad \quad \quad \text{j} \\
 & \quad \quad \quad \dots 1 \quad \quad \quad t - n, k, \lambda \\
 & \quad \quad \quad \text{m} \quad \text{m} \quad \quad \quad \text{k} \quad \quad \quad d_{\min} \quad \text{k} \quad \deg f \\
 & \quad \quad \quad \text{k} \quad \quad \quad \text{m} \quad \text{m} \quad \quad \quad \text{m} \quad \text{k} \\
 & f(x) \quad \text{m} \quad x \in \mathbb{F}_2^n \\
 & \text{supp } x
 \end{aligned}$$

**Example 2** The ANF  $f(x_3, x_2, x_1) = x_1x_2 + x_1x_3 + x_2x_3$  is associated with a  $(3, 1, 1)$  (complete) symmetric design with  $\lambda = -1, \mu = -1, \nu = -1, \omega = -1$ . Then  $f$  is balanced since  $\text{supp } f$  contains the three blocks (monomials).

$$\text{m} \quad \text{m} \quad \cdot$$

**Proposition 3** In a  $t - n, k, \lambda$  SBIBD, with  $n = k$ , for all  $b_i, b_j \in \mathbb{F}_2^n$  we have

$$\begin{aligned}
 & b_i - b_j \in \mathbb{F}_2^{n-1} \\
 & \text{Proof.} \quad b_i - b_j \in \mathbb{F}_2^{n-1} \quad \text{mm} \quad b_i - b_j \in \mathbb{F}_2^{n-1} \\
 & b_i - b_j \in \mathbb{F}_2^{n-1} \quad \lambda \quad k - n. \quad \text{mm} \\
 & \quad \quad \quad \lambda \quad \frac{k(k-1)}{n-1} \quad \text{k} \quad \text{k} \\
 & k^2 - n - 1 \cdot k \quad n^2 - n \quad \text{k} \quad n - 1.
 \end{aligned}$$

**Remark :**  $n, n-1, n-1, n-1$  mm

$$\text{m} \quad \text{m}$$

**Proposition 4** Let  $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  a Boolean function associated with an  $(n, n-1, n-1)$  symmetric design. Then  $f$  is never balanced except for  $n = 1$ , and its weight is given by :

$$\text{wt } f = \begin{cases} n & \text{if } n \text{ even} \\ n-1 & \text{if } n \text{ odd} \end{cases}$$

*Proof.*

*Proof.*

$$\lambda - \frac{1}{m} = \frac{\lambda m - 1}{m}.$$

*Remark :*

$$n^{-1} = \frac{1}{m} \cdot \frac{m}{m}.$$
$$wt\, f < n^{-1}$$





$$\begin{aligned}
 & -\alpha - f \supp \alpha \quad A. \\
 & -A - f. \quad m \quad \alpha \quad f \quad x \quad 1 \quad \supp x \quad A \\
 & \quad m \quad \alpha - f \quad \frac{n+1}{2} \quad \frac{n}{2} \quad n \\
 & \quad \quad \quad f \quad x \quad 1 \quad \supp x \quad A. \\
 & \text{eov} \left( \alpha \quad I \equiv 1 \quad m \right. \\
 & -f \rightarrow -\alpha - \beta - \\
 & I_{\alpha,\beta}, I_{\alpha,\delta} \quad I_{\beta,\delta} \quad i \\
 & \quad \quad \quad I_{\alpha,\beta}, I_{\alpha,\delta}, I_{\beta,\delta} \quad I_{\alpha,\beta,\delta} \\
 & \quad \quad \quad I_{\alpha,\beta,\delta} \\
 & \quad \quad \quad m
 \end{aligned}$$

$$I_{\alpha,\beta,\delta} - I_{\alpha,\beta} - I_{\alpha,\delta} - I_{\beta,\delta} - \left( I_{i,j} - I_{j,k} - I_{\alpha,\beta} - I_{\alpha,\delta} - I_{\beta,\delta} - \right.$$

$$i \quad m \quad m \quad -f \quad i > \quad mm \quad . \\
 \quad \quad \quad MAJ_n$$

$$x \quad wt \quad x = d_{min} \quad f \quad x \quad 1.$$

Remark :

$$\begin{aligned}
 & \quad m \quad m \quad q^{-1} \quad MAJ_{2^q-1} \quad . \\
 & \quad m \quad m \quad m \quad x \\
 & m \quad x \quad m \quad k \quad .
 \end{aligned}$$

$$\begin{aligned}
 & -f \quad \text{union property} \quad F - F = \mathbb{N}_n, \quad -F - -, \quad -F - - \quad m \\
 & \quad k \quad m \quad m \quad 1 \\
 & - \prec \quad n-2.
 \end{aligned}$$

$$\begin{aligned}
 & \quad \quad \quad MAJ_n \\
 & x \quad m \quad n.
 \end{aligned}$$

**Proposition 7** *The Boolean functions  $MAJ_n$  have correlation order 1 and*

$$P \ MAJ_n \ x \quad x_i \quad \frac{1}{n} \quad \frac{\binom{n-1}{\frac{n-1}{2}}}{n}$$

*Proof.*

$$\begin{aligned}
 & \quad \quad \quad MAJ_n \ x \quad MAJ_n \ x \quad \overline{1} \quad 1 \\
 & \quad \quad \quad m \\
 & \quad \quad \quad m \quad m \quad m \\
 & \quad \quad \quad m \quad n \\
 & \quad \quad \quad 1 \quad P \ MAJ_n \ x \quad x_i = \frac{1}{2}. \quad m \\
 & -x - \mathbb{F}_2^n \ MAJ_n \ x \quad x_i - \quad m \quad i.
 \end{aligned}$$



	1	1	
			<i>Balanced Boolean Functions</i>
		1	
			<i>Balance testing and balance-testable design of logic</i>
<i>circuits</i>	1	1	
			<i>Cumulative balance testing of logic circuits</i>
	1		<i>Decoding of cyclic codes and codes on curves</i>
		1	
			11
	1		
			1
			<i>Construction of Bent Functions and Balanced Boolean Functions</i>
<i>with High Nonlinearity</i>			
		1	
			1
	1	1	<i>Highly Nonlinear Balanced Boolean Functions with a good</i>
<i>Correlation-Immunity</i>			<i>Correlation-Immunity</i>
		1	
			1
	1		<i>Extremal Set Systems</i>
"	1		
		)	1
			1
			1
	11		
		1	
11			11
			11
	1		
			11
<i>Characteristics of Boolean Functions</i>			<i>Propagation</i>
		1	
			1
	1		<i>Correlation Immunity of Nonlinear Combining Functions for</i>
<i>Cryptographic Applications</i>			<i>Cryptographic Applications</i>
			1
			<i>Partial Geometries</i>
	1		
			1
			<i>Restriction, Terms and Nonlinearity of Boolean</i>
<i>Functions</i>		1	
			1

# Coding Applications in Satellite Communication Systems

## [Invited Paper]

Dr Sean McGrath

University of Limerick  
Ireland  
sean.mcgrath@ul.ie

**Abstract.** This paper provides a brief insight in satellite communication systems from the perspective of coding applications. CDMA based systems for use in Low Earth Orbit (LEO) satellite systems is the focus of the paper. The code-division-multiple-access (CDMA) format is emerging as a dominant air interface technology for cellular, personal-communications-services (PCS) as well as satellite installations. This transmission technology relies on a combination of spread-spectrum modulation, Walsh coding, and sophisticated power-control techniques. In a typical CDMA transmitter, a data signal is encoded using a Walsh code and then mixed with the RF carrier, which has been spread using a pseudorandom-noise (PN) source. In a base-station transmitter, multiple data signals are assigned unique Walsh codes and combined. In the CDMA receiver, the signal is filtered and fed to a correlator, where it is despread and digitally filtered to extract the Walsh code. The paper examines some weaknesses of such systems.

## LEO System

The systems of Low-earth orbiting (LEO) satellites provide mobile voice, data and facsimile and other mobile satellite services for both domestic and international subscribers. The systems consists typically consist of a space segment, a user segment and a Ground segment, which connects to the terrestrial telephone network. The space segment consists of any thing from 10 to 66 satellites orbiting the earth at an altitude of over 1000Km. The user segment is composed of hand-held, mobile and fixed terminals. The ground segment consists of the satellite control center and Gateways. The systems of Low-earth orbiting (LEO) satellites provide mobile voice, data and facsimile and other mobile satellite services for both domestic and international subscribers. The systems consists typically consist of a space segment, a user segment and a Ground segment, which connects to the terrestrial telephone network. The space segment consists of any thing from 10 to 66 satellites orbiting the earth at an altitude of over 1000Km. The user segment is composed of hand-held, mobile and fixed terminals. The ground segment consists of the satellite control center and Gateways.

## CDMA

A CDMA spread spectrum signal is created by modulating the radio frequency signal with a spreading sequence known as a pseudo-noise (PN) digital signal because they make the signal appear wide band and "noise like". The PN code runs at a higher rate than the RF signal and determines the actual transmission bandwidth. Messages can also be cryptographically encoded to any level of secrecy desired with direct sequencing as the entire transmitted/received message is purely digital. An SS receiver uses a locally generated replica pseudo noise code and a receiver correlator to separate only the desired coded information from all possible signals. A SS correlator can be thought of as a specially matched filter -- it responds only to signals that are encoded with a pseudo noise code that matches its own code. Thus an SS correlator (SS signal demodulator) can be "tuned" to different codes simply by changing its local code. This correlator does not respond to man made, natural or artificial noise or interference. It responds only to SS signals with identical matched signal characteristics and encoded with the identical pseudo noise code.

## Air-Interface

CDMA was selected due to its interference tolerance as well as its security inherent in the modulation scheme. CDMA is able to provide good voice quality while operating at relatively low RF power levels. Path diversity is employed using rake receivers to receive and combine the signals from multiple sources. In the forward direction the use of diversity brings substantial gain if one of the satellites is obstructed. However, the reverse direction, because this is non-coherent diversity combining the gain is not as good.

Assignment of the code channels transmitted by a gateway. Out of the 128 code channels the forward channel consist of pilot channel, one sync channel, up to seven paging channels, and a number of Forward Traffic Channels. Multiple Forward channels are used in a Gateway by placing each Forward channel on a different frequency, namely the forward link pilot, sync and paging channel.

The pilot channel will generate an all zeros Walsh Code. This combined with the short code used to separate signals from different Gateways and different satellites. The pilot channel is modulo 2 added to 1.2288 Mc/s short code and is then QPSK spread across the 1.23 MHz CDMA bandwidth.

The Sync Channel is interleaved, spread and modulated spread spectrum signal. The sync channel will generate a 1200b/s data stream that includes time, gateway identification and assigned paging channel. This convolutionally encoded and block Interleaved to combat fast fading. The resulting 4800 symbols per second data stream is modulo two added to the sync Walsh code at 1.2288Mc/s which is then modulated using QPSK across the 1.23MHz CDMA bandwidth.

The paging Channel is used to transmit control information to the user terminal. The paging channel is convolutionally encoded at rate  $=1/2$ , constraint length  $K = 9$

and block interleaving. The resulting symbol rate is combined with the long code. The paging channel and long code are modulo two added, which is then modulo two added to the 1.2288Mc/s Walsh Code.

## **Modulation & Spreading**

The spreading sequence structure for a CDMA channel comprised on an inner PN sequence pair and a single outer PN sequence. The inner PN sequence has a chip rate of 1.2288 Mcps and a length of 1024, while the outer PN sequence has an outer chip rate of 1200 outer chips per second and a length of 288. The outer PN sequence modulates the inner PN sequence to produce the actual spreading sequence lasting 240msec. Exactly one inner PN period is contained within a single outer PN chip.

Other parameters such as Link delay are important end-to-end parameters. The LEO satellites will provide a much more benign delay than the more common synchronous orbit satellites. Delay is held to 150ms in each direction. The vocoder uses a Code Excited Linear Prediction (CELP) algorithm which is similar to that used by the IS-96 coder.

## **Conclusion**

This paper provides an insight and overview LEO systems and the application of this current area and the possible next generation systems. Underlining focus was on coding used in satellite applications and in particular to CDMA systems. The discussion has involved coding in all aspects of the satellite system, from user terminals to ground stations are discussed. The implementation of the various blocks are discussed. Finally the paper looks at future satellite systems and examines the coding requirements.

# A Unified Code

Xian Liu<sup>1</sup>, Patrick Farrell<sup>2</sup>, and Colin Boyd<sup>3</sup>

<sup>1</sup> Communications Research Group, School of Engineering, University of Manchester,  
Manchester M13 9PL, UK

`mbgeexl2@fs1.eng.man.ac.uk`

<sup>2</sup> Communications Research Centre, Lancaster University, Lancaster LA1 4YR, UK  
`p.g.farrell@lancaster.ac.uk`

<sup>3</sup> School of Data Communications, Queensland University of Technology, Brisbane  
Q4001, Australia  
`boyd@fit.qut.edu.au`

**Abstract.** We have proposed a novel scheme based on arithmetic coding, an optimal data compression algorithm in the sense of shortest length coding. Our scheme can provide encryption, data compression, and error detection, all together in a one-pass operation. The key size used is 248 bits. The scheme can resist existing attacks on arithmetic coding encryption algorithms. A general approach to attacking this scheme on data secrecy is difficult. The statistical properties of the scheme are very good and the scheme is easily manageable in software. The compression ratio for this scheme is only 2 % worse than the original arithmetic coding algorithm. As to error detection capabilities, the scheme can detect almost all patterns of errors inserted from the channel, regardless of the error probabilities, and at the same time it can provide both encryption and data compression.

## 1 Introduction

Data compression, cryptologic algorithms, and error control coding are the central applications in information theory and are the key activities in a communication system. In fact, efficiency and reliability are the main concerns in a communication system. Data compression increases the efficiency by reducing the transmission and storing sizes without losing information significantly; cryptologic algorithms denies the unauthorised users trying to read or modify the messages being transmitted or stored; error control coding provides protection against channel errors. For error control, there are two basic strategies: forward error correcting (FEC) and automatic repeat request (ARQ). FEC works with an appropriate error correcting code and can, within the code's ability, automatically recover the inverted bits resulting from channel errors at the receiver end. ARQ applies a suitable error detecting code so that the decoder at the receiver end is within the code capability able to detect if the encoded file received has been damaged by channel errors and request the sender to retransmit the file. The essential fact in error control coding is that appropriate redundancy is introduced in the encoded file.

Arithmetic coding provides an effective mechanism for removing redundancy in the encoding of data. It can achieve theoretical compression ratio bounds so it has gained widespread acceptance as an optimal data compression algorithm. The first practical implementation for arithmetic coding was provided by Witten, Neal, and Cleary [1, 12] in 1987 (which is called the WNC implementation in this paper). Since then, many different implementations of arithmetic coding with different models have appeared. The authors of this paper have investigated the possibilities of providing cryptology and error control based on arithmetic coding and proposed a scheme providing both encryption and data compression [8], a scheme providing both error correction and data compression [10], and a scheme providing encryption, data integrity, and data compression, all together in a one-pass operation [9]. In this paper we will propose a unified code that can provide encryption, error detection, and data compression all together in a one-pass operation. The efficiencies in both encryption and data compression are the same as our previous schemes but also the scheme can detect almost all error patterns inserted from the channel, regardless of the error probabilities.

## 2 Arithmetic Coding

Arithmetic coding is based on the fact that the cumulative probability of a sequence of statistically independent source symbols equals the product of the source symbol probabilities. In arithmetic coding each symbol in the message is assigned a distinct subinterval of the unit interval of length equal to its probability. This is the encoding interval for that symbol. As encoding proceeds, a nest of subintervals is defined. Each successive subinterval is defined by reducing the previous subinterval in proportion to the current symbol's probability. When the message becomes longer, the subinterval needed to represent it becomes smaller, and the number of bits needed to indicate that subinterval grows. The more likely symbols reduce the subinterval by less than the unlikely symbols and thus add fewer bits to the message. This results in data compression. When all symbols have been encoded, the final interval has length equal to the product of all the symbol probabilities and can be transmitted by sending any number belonging to the final interval. That means if the probability of the occurrence of a message is  $p$ , arithmetic coding can encode that message in  $-\log_2 p$  bits, which is optimal in the sense of the shortest length encoding. The pseudocode of arithmetic coding is as follows:

```
/* In the model, symbols are numbered 1, 2, 3, ... */
/* The cum_prob[ ] stores the                               */
/* cumulative probabilities of symbols with                   */
/* cum_prob[i] increasing as i                               */
/* decreases and cum_prob[0]=1. The encoding                */
/* transmits any value in the final [low, high)              */
/* when it is finished.                                     */
```

```

Encoding: The initial encoding interval [low, high) = [0, 1)
EncodeSymbol (symbol, cum_prob)
range = high - low;
high = low + range * cum_prob[symbol-1];
low = low + range * cum_prob[symbol];

Decoding: The initial decoding interval [low, high) = [0, 1)
DecodeSymbol (cum_prob)
find symbol such that cum_prob[symbol]
    <= (value - low)/(high - low)
    < cum_prob[symbol - 1];
range = high - low;
high = low + range * cum_prob[symbol - 1];
low = low + range * cum_prob[symbol];
return symbol;

```

The WNC implementation for arithmetic coding [1, 12] was the first practical algorithm and is widely accepted. The algorithm is provided with either a static model or a first-order adaptive model. The algorithm realises integer arithmetic and incremental transmission. The arithmetic precision is 16-bit. In their first-order adaptive model, all the frequencies are initialised to 1. If the current model exceeds the maximum cumulative frequency, the model reduces all frequencies by half and recalculates cumulative frequencies. If necessary the model reorders the symbols to always put the current one in its correct rank in the frequency ordering. Adaptation is performed by incrementing the proper frequency count and adjusting cumulative frequencies accordingly. Due to the limit of the first-order adaptation, the compression ratio is 50% to 70% according to the size and type of the file. However it can be greatly improved by using a higher-order adaptive model.

### 3 Our Basic Scheme

Of course, the purpose of data compression is to reduce the redundancy in the message. On the other hand, redundancy contained in the output of a cryptosystem is usually one of the main resources to be used by the cryptanalyst. Also in an adaptive model, the current state in the model is related to the initial state and all of the messages that have been encoded so far since the model was initialised. Based on these facts Witten et al [11] suggested that an adaptive arithmetic coding algorithm may provide high level security. They also indicated that to use an adaptive modelling compression algorithm as an encryption algorithm it was enough to transmit the initial state in the model as a key over a secure channel.

#### 3.1 Witten-Cleary Proposal

In 1988 [8] Witten and Cleary suggested two ways to insert the key into arithmetic coding:

**Method 1:** The initial model is used as the key in which an array of single-character frequencies in the range of 1-10 would do.

**Method 2:** A constant initial model is used and before transmission begins both the encoder and decoder assimilate a short secret message into the model.

Their further suggestion is that the adaptive links should be maintained over long periods of time; i.e. the final model of encoding the current message will become the initial model to encode the next message.

Aiming at Method 1 in Witten-Cleary proposal with WNC adaptive implementation, Bergen and Hogan suggested a chosen plaintext attack on first-order adaptive arithmetic coding in 1993 [2]. Instead of trying to recover the initial model the Bergen-Hogan attack tries to take control of the model and reduce it to a manageable form. If the encoder does not initialise its model, the attacker can decrypt any message transmitted after the attack is done. To be successful, in the Bergen-Hogan attack an associate as well as an attacker are necessary. The associate needs to send  $2^{18}$  symbols and the attacker needs to try decoding the test string  $2^{14}$  times. Up until now the Bergen-Hogan attack is the only feasible attack on the adaptive arithmetic coding encryption algorithm.

### 3.2 Our Basic Scheme

In [8, 9] we proposed our initial scheme to provide both encryption and data compression, and the scheme to provide encryption, data integrity, and data compression all together in a one-pass operation. In this section we will summarise some of the related results and the further results on the updated scheme providing both encryption and data compression. The key point is that we found without any exception that if the state in the adaptive model in the decoder is only slightly different from that in the encoder the decoder is unable to work at all and if the current interval in the decoder is only slightly different from that in the encoder the decoder is also unable to work at all.

#### The Scheme

1. Select an initial frequency count for every symbol randomly, which acts as the initial state in the model. The only restriction is that these numbers are all larger than 0.
2. Select the initial interval within the full range randomly, but the length of the initial interval should not be less than  $(2^{16} - 1)/4$ .
3. Select a secret 16-bit substitution with key size 16 bits, which is used to substitute for the first 16-bit output of the encoding.
4. Choose two secret parameter pairs  $(\varepsilon_l^0, \varepsilon_h^0)$  and  $(\varepsilon_l^1, \varepsilon_h^1)$  which are used to shrink the current interval controlled by a random 64-bit string cyclically, where the four parameters are different.

### 3.3 The Key Size

Firstly, 96 bits can be used to indicate the initial state. There are 96 symbols with exact meaning in the extended ASCII set in text compression, so 96 bits

are enough to indicate the initial state. If the bit is 0 set the count of the symbol to 2, otherwise to 3. Set the counts of the remaining 160 symbols to be 1. The total number of different initial states in the model is  $2^{96}$ . Secondly, 32 bits can be used to indicate the initial interval. The initial interval can be indicated by determining *low* and *high*, or *low* and *range*, so 16 bits are used to indicate one of them and 32 bits for both. The number of all of the valid initial intervals is  $2^{30}$ . Thirdly, 40 bits can indicate the shrinking parameters.  $\varepsilon_l^i$  is chosen to be 4 decimal digits in the form of 0.0\*\*\* and  $\varepsilon_h^i$  is chosen to be 4 decimal digits in the form of 0.9\*\*\*. The unknown part in every parameter ranges from 0 to 999, so 10 bits are necessary to indicate each of them. And then, we use 64 bits for the random control string. Finally, 16 bits are used for the 16-bit substitution key size. A secret 16-bit substitution with key size 16 bits is preferable. So the total key size for the scheme is 248 bits. It should also be pointed out that in our basic scheme, the second shrinking for the *low* is based on the new interval resulting from the first shrinking for the *high*.

### 3.4 Communication Protocol

**Protocol 1:** The final state in the model during encoding of the current message becomes the initial state of the model to encode the next message, and during the lifetime of using the same key the model will be initialised regularly. When encoding the current message is finished, shift the cyclic shift register storing the 64-bit random string one step and the next bit will become the first control bit to control the first shrinking in encoding the next message. Whenever finishing the encoding of the current message, all of the rest will be initialised.

This protocol has the advantage that the initial model to encode the next message is relevant to all of the messages which have been sent since initialisation.

**Protocol 2:** The only change compared with protocol 1 is that whenever encoding is finished, the model is re-initialised.

Protocol 2 on its own definitely denies the Berger-Hogan attack because the attack is unable to find the initial state in the model. Until now the Bergen-Hogan attack is the only powerful approach to attacking arithmetic coding encryption algorithms, so protocol 2 is preferable.

### 3.5 The Strength

In the Bergen-Hogan attack the attacker knows he matches the keying materials only when he successfully decodes the test string. To use the Bergen-Hogan attack on our proposal, the associate's strategy is the same as that to attack with the Witten-Cleary proposal, but the attacker's work will be increased dramatically. In order to decode the test string the attacker has to find the first symbol's frequency count in the standard form, the initial interval, the substitution, the pairs  $(\varepsilon_l^0, \varepsilon_h^0)$  and  $(\varepsilon_l^1, \varepsilon_h^1)$ , and the 64-bit control string all together, instead of just trying the first symbol's frequency count in the standard form  $2^{14}$  times in breaking with the Witten-Cleary proposal. The attacker has to try to decode the test string  $2^{14} \cdot 2^{30} \cdot 2^{16} \cdot 2^{19} \cdot 2^{19} \cdot 2^{64} = 2^{168}$  times. Partially finding

the keying materials is also very difficult. The reasonably simplest way for the attacker would be to firstly find the first symbol's frequency count in the standard form together with the initial interval. For this purpose the attacker only needs to decode the first symbol in the test string, but he has to try decoding  $2^{14} \cdot 2^{30} \cdot 2^{16} \cdot 2^{19} = 2^{79}$  times.

It has been shown in [8, 9] that compared with the WNC first order adaptive implementation, the compression ratio of our scheme is only 2% worse and the running time is slightly less than double of that of the WNC implementation. Furthermore, the encoded files with our scheme have very good randomness. Changing any number of bits in the file to be encoded results in the fact that from the position in the encoded file that the first changed bit corresponds to, then in the subsequent output, if this encoded file is compared with the encoded file resulting from the totally unchanged original file to be encoded, the changed bits and unchanged bits take the probabilities 0.5, and distribute uniformly and randomly. Furthermore, the outputs of our scheme and the files of the bitwise modulo 2 addition of the output of our scheme and the outputs of our scheme with the key being changed randomly, as well as the file of the bitwise modulo 2 addition of the output of our scheme with the file to be encoded in which one bit near the beginning was inverted and the output of our scheme with the unchanged file to be encoded have passed the frequency test, the binary derivative test, the change point test, the poker test, the runs test, the sequence complexity test, the linear complexity test, and Maurer's universal test (the statistical test software Crypt-X [5] is from the Information Security Centre at the Queensland University of Technology), and also there is no statistical difference between the output from our modified scheme and that from the DES. So good plaintext diffusion and ciphertext avalanche are achieved. Also, in the modified scheme, the keying materials have very good effects on balance, diffusion, completeness, and avalanche.

It has also been shown in [8, 9] that a general approach to attacking our scheme is difficult and our scheme can resist other related attacks [4, 6] to arithmetic coding encryption algorithms. In fact, if the model our scheme works with is a fixed binary model with known symbol probabilities, and the initial substitution is ignored, and also the scheme works with theoretical arithmetic coding, finding the key is equivalent to solving two polynomial equations with 70 variables and degree 256. However, this analysis is based on a much simplified fixed binary model and with ideal theoretical arithmetic coding. Practically, our scheme works with the Witten-Cleary first order adaptive model with alphabet size 256. So far, there has not been any method to trace the evolution of the adaptive model. Also our scheme works with Witten-Cleary implementation for arithmetic coding. Quite a few practical strategies in the implementation make the coding procedure much more difficult to trace than in theoretical arithmetic coding. In fact, there are a number of main differences between the scheme with theoretical arithmetic coding with a fixed binary model and the scheme with WNC first order adaptive arithmetic coding. Firstly, for the WNC first adaptive arithmetic coding, there is no way to find the exact current state in the

adaptive model. One may argue that as the uncertainty of the current state in the adaptive model, if a chosen plaintext attack is used and the secret shrinking parameters, the initial interval, and the secret control string as well are known, totally depends on the secret initial state in the model, after encoding a huge known file the effect from the initial model is trivial; i.e., it can be converted from any known initial model. However, this is not true with WNC first order adaptive model. Approximation does not make any sense unless the two current states are the same in arithmetic coding. Secondly, WNC adaptive arithmetic coding uses 16-bit finite precision. That means it has to expand the current interval after encoding (decoding) one symbol. Such expansions are unpredictable and untraceable with our scheme. Therefore, such a regular relation between the input and the output in the encoder definitely does not hold in our scheme with WNC adaptive arithmetic coding. One thing clear is that if a mathematical relation exists in WNC adaptive arithmetic coding it must be much more complicated.

## 4 A Scheme Providing Encryption, Error Detection, and Data Compression All Together

In arithmetic coding, the decoding is successful only when the current interval in the decoding is identical to that in the encoding and the current state in decoder's model is the same as that in encoder's model. In case the current interval or the current state in the model in the decoder is not the same as that in the encoder, the whole subsequent encoded file would be unable to be recovered. That is, even a single bit error appearing in transmission would probability corrupt the remainder of the file. So the problem with the compressed data is that it is highly susceptible to transmission errors. The better the compression achieved, the more serious the effect will be. In practice, error control techniques will have to be used to prevent transmission errors and to provide arithmetic coding with a completely noise free channel.

In fact, it is not difficult to introduce redundancy into arithmetic coding. An obvious way to introduce redundancy into arithmetic coding is to shrink the current interval in some way: after encoding a symbol or periodically. This method is equivalent to the method demonstrated as follows, because to add the same amount of redundancy periodically is the same as to encode an extra symbol with fixed probability periodically. Besides the adaptive compression model, we use an additional fixed model, in which there are only two symbols: the check symbol and the forbidden symbol. The check symbol is encoded periodically after one or several symbols from the adaptive compression model is encoded, and the forbidden symbol is never encoded. In decoding, the compression and check models are used alternately. If the forbidden symbol is decoded, an error has occurred. Redundancy can be controlled by varying the probability of the check symbol. Like convolutional codes, the redundancy is spread evenly through the message and errors may be detected soon after they occur. The error control performance will be related to how frequent the check symbol is encoded as well

as how much the current interval is shrunk or the probability of the check symbol arranged in the fixed check model.

#### 4.1 The Scheme

The third author of this paper proposed a scheme [3] which provides both error detection and data compression. The strategies he used are to shrink the current interval by a fixed amount after encoding (decoding) every symbol, to add an extra final check symbol after encoding (decoding) the EOF symbol to detect errors at the end of the file, and to exclusive-or every adjacent pair of output bits to be able to detect single isolated error bits. It was declared in [3] that the scheme could detect all single isolated error bits.

In order to provide error detection in our basic scheme as well, it is definitely necessary to introduce some kind of redundancy into the encoding procedure. In our basic scheme we shrink the current interval twice after encoding (decoding) a symbol, which is one of our main steps to insert the secret key. It is our opinion that we have already added appropriate redundancy in the coding procedure even though it was not intended for error detection. We also think that it is not necessary to exclusive-or adjacent pair of output bits, but we do need to encode an extra final check symbol at the end of each file.

Compared with our basic scheme we only need to encode an extra final check symbol after encoding the EOF symbol for each file to achieve encryption, data compression, and error detection all together in a one-pass operation, with almost no extra price.

#### 4.2 Performance

We have done exhaustive tests for the performance of the error detection abilities of our scheme. The decoder always succeeds in detecting the first error bit and stops after a short delay. We have also found that the extra final check symbol is very effective in detecting any errors at the end of the file. The error detection performance of this scheme is independent of the error probabilities.

In the formal test, the file tested is book1.html from the Calgary Corpus with size 768771 bytes, which is a typical English technical report, and the compressed size is 451729 bytes; the compression ratio is 59%. Random errors are inserted to construct four groups of error probabilities:  $i/1000000$   $i = 1, 2, \dots, 999$ ,  $i/100000$   $i = 1, 2, \dots, 999$ ,  $i/10000$   $i = 1, 2, \dots, 999$ , and  $i/1000$   $i = 1, 2, \dots, 1000$ , which result in about 4000 different values of error probability ranging from  $10^{-6}$ , step by step, to 100%. According to these error probabilities, random errors are inserted into the encoded file, resulting in about 4000 error corrupted encoded files. The decoder succeeds in detecting the first error bit in all of the 4000 files after a short delay. The mean value of the delay is 92.88 bits and the standard deviation of the delay is 95.16 bits. That means, in the exhaustive test, the decoder always finds the first error inserted after a delay of around 92.88 bits.

The experimental results from the exhaustive test allow us to predict that the scheme can detect most if not all error patterns inserted from the channel, independent of the error probabilities. This is in contrast to the fact that in the original arithmetic coding scheme the decoder cannot usually detect errors.

It is necessary to compare the performance of our scheme with the performances of dedicated standard error detection codes. Traditionally, the dedicated error detection codes are cyclic redundancy check codes (CRC), such as IEC TC57, IEEE WG77.1, ANSI, IBM-SDLC, and CCITT X.25, because cyclic codes are very effective in detecting burst errors [7].

The CCITT X.25 CRC code has the generator polynomial:

$$G(x) = x^{16} + x^{15} + x^{12} + 1.$$

The minimum distance of this code is 4. In a block length up to 32768 bits it can detect: all triple or fewer random errors, all odd numbers of errors, all bursts of length up to 16, and 99% of all other longer bursts. However, there are many combinations of error patterns of even weight that the code cannot detect and that means it can only detect 50% of all of errors. So the error detection performance of our scheme is even better than that of the dedicated standard cyclic redundancy check codes. However, in addition, our scheme can provide both encryption and data compression.

## 5 Conclusions

In this paper we present a scheme that can provide data encryption, data compression and error detection all together in a one-pass operation. The scheme is based on WNC implementation for arithmetic coding in which a first order adaptive model is used. The total key size is 248 bits. The statistical properties of our scheme are very good. Attacking this scheme is difficult. The compression ratio is about 2% worse than WNC implementation for arithmetic coding. The scheme can detect almost all patterns of errors inserted from the channel, independent of the error probabilities.

## References

1. Bell T., Cleary J., and Witten I.: *Text compression*, Prentice Hall, 1990.
2. Bergen H. and Hogan J.: "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm", *Computers and Security*, Vol.12, 1993, pp.157-167.
3. Boyd C., Cleary J., Irvine S., Rinsma-Melchert I., and Witten I.: "Integrating error detection into arithmetic coding", *IEEE Trans. COM*, Vol.45, No.1, 1997, pp.1-3.
4. Cleary J., Irvine S., and Rinsma-Melchert I.: "On the insecurity of arithmetic coding", *Computers and Security*, Vol.14, 1995, pp.167-180.
5. Crypt-X, Statistical Package Manual, Measuring the Strength of Stream and Block Ciphers. Information Security Research Centre, Queensland University of Technology, 1990.

6. Irvine S. and Cleary J.: "The subset sum problem and arithmetic coding", private communication, 1995.
7. Klove T. and Korzhik V.: *Error Detecting Codes, General theory and their application in feedback communication systems*, Kluwer Academic Publishers, 1995.
8. Liu X., Farrell P., and Boyd C.: "Resisting the Bergen-Hogan attack on adaptive arithmetic coding", LNCS-1355, Cryptography and Coding, Springer, December, 1997, pp.199-208.
9. Liu X., Farrell P., and Boyd C.: "Arithmetic coding and data integrity", Proceedings of WCC'99, pp.291-299, Paris, 11th-14th January, 1999.
10. Liu X. and Farrell P.: "Arithmetic coding with error correction", Proceedings of PREP'99, pp.330-333, Manchester, 5th-7th January, 1999.
11. Witten I. and Cleary J.: "On the privacy afforded by adaptive text compression", Computers and Security, Vol.7, 1988, pp.397-408.
12. Witten I., Neal R. and Cleary J.: "Arithmetic coding for data compression", Communications of the ACM, Vol.30, No.6, 1987, pp.520-540.

# Enhanced Image Coding for Noisy Channels

Paul Chippendale, Cagri Tanriover, Bahram Honary

Department of Communication Systems  
Lancaster University, Lancaster LA1 4YR, United Kingdom  
[p.chippendale, c.tanriover, b.honary}@lancs.ac.uk](mailto:{p.chippendale, c.tanriover, b.honary}@lancs.ac.uk)  
<http://www.dcs.lancs.ac.uk>

**Abstract.** This paper explores the application of a combined error resilient coding scheme to image transmission over time-varying noisy channels. To improve performance at low signal-to-noise ratios, turbo coding is incorporated into the system. Demonstrated through simulations, this novel combination of source and channel coding is shown to correct and restrict errors incurred during transmission over Additive White Gaussian Noise (AWGN) and Rayleigh Fading channels. The error correcting capability of the coding scheme also illustrated with compressed and uncompressed image transmission results which are comparable in terms of their visual quality.

## APEL Coding

Absolute addressed Picture ELe ment coding (APEL) [1], [2] is a lossless, robust image coding system which translates variable sized pixel areas of pre-defined dimensions into independent picture blocks (pels). Each pel is issued with two co-ordinates,  $x$  and  $y$ , establishing an absolute location with respect to an origin.

As the APEL coding technique operates on a binary level, the encoding of  $n$ -bit grey-scale or colour images employs a Bit Plane Coding (BPC) [2] stage. The BPC stage furnishes the APEL encoder with a colour coding sequence to represent a given source image in  $n$  binary planes.

Taking each extrapolated binary plane in turn, a recognition algorithm searches through each image looking for square areas of black pixels; starting with large square pels during the first scan, then repeating this process in multiple passes selecting pels of decreasing magnitude. The maximum size of the initial pel is limited according to the anticipated nature of the channel, consequently less information is lost should corruption occur. Once all of the square pels of an efficient size have been removed from the plane, run-lengths of various geometries are used to encode the residue. Fig. 1 illustrates an APEL encoded section of an image. Here, it can be seen how  $(x,y)$  co-ordinates are assigned to pels of various geometries.

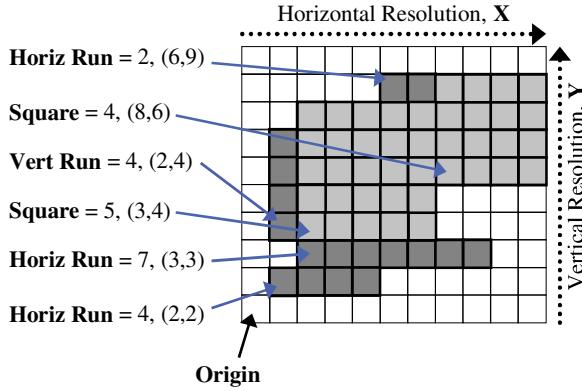


Fig. 1. APEL coded section of an image

The data-stream created from this process can be pictured as a succession of  $(x,y)$  addresses, grouped according to pel size and interspersed with control symbols (see Fig. 2). These symbols not only serve to provide synchronisation markers, but in addition convey pel geometry metrics to the decoder [2].

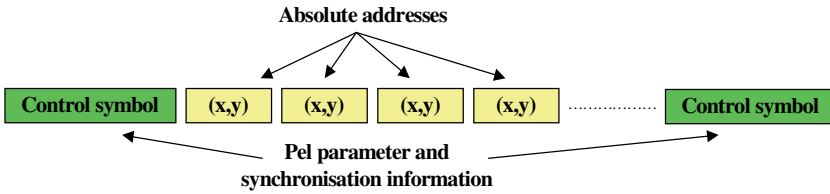


Fig. 2. Breakdown of APEL data stream

The APEL scheme alleviates the need for End Of Line (EOL) symbols and, as each codeword is independent, offers a solution to the problems of horizontal and vertical error propagation. Additionally, as each pel has its own address, it is possible to interleave them within the transmitted data-stream. This versatility can be utilised in many ways, for example: pels pertaining to important image detail can either be dispersed throughout the data-stream or transmitted at the start depending on channel conditions or operator preference.

## Application of Turbo Coding to APEL

Turbo codes [3] are forward error correction schemes which employ concatenated component codes, interleaving and iterative decoding principles to achieve bit error rate performance close to the Shannon limit. Decoding is performed by the sub-optimal log-Maximum Aposteriori Probability (MAP) algorithm [4], which improves

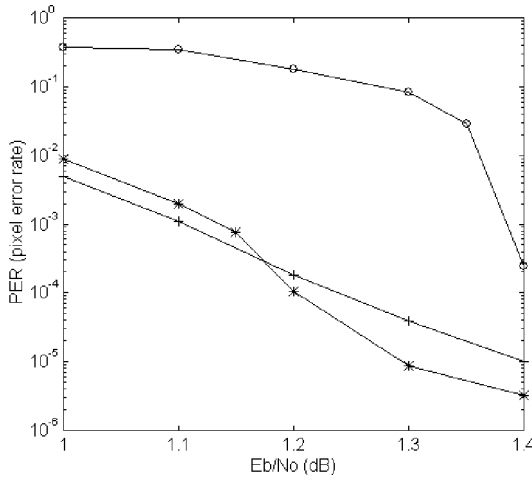
the accuracy of the decoded information symbols through a set of iterations where soft extrinsic information is passed between the component decoders.

In this paper, a turbo encoder with parallel concatenation is incorporated into an APEL system. The turbo codec implemented is composed of two recursive systematic convolutional component codes, of rate  $\frac{1}{2}$  and constraint length 3. In general, the use of systematic convolutional codes provides robustness against decoding errors by decreasing the minimum free distance of the code. As a consequence of minimising the free distance, the error correction capability of the system improves. It is this feature of systematic convolutional codes that prevents catastrophic error propagation.

It should also be noted that the turbo decoder used in this coding scheme is designed to correct random errors only, hence the majority of burst errors encountered during transmission cannot be corrected.

## Results

To demonstrate the benefits gained and also to provide benchmarks for comparison, in addition to incorporating turbo coding into an APEL system, we also concatenated the aforementioned turbo codec onto JPEG [5] and bitmap (BMP) file formats. Simulations over AWGN channel, at various signal-to-noise ratios, attest to the excellent performance of the APEL-turbo combination compared to the application of turbo coding onto JPEG and BMP file formats. The results presented in this paper were obtained from simulations conducted using an interleaver length of 8000 bits and performing 16 decoding iterations.



**Fig. 3.** Turbo coded image transmission over AWGN channel *Turbo coded JPEG* (o), *Turbo coded BMP* (+), *Turbo coded APEL* (\*)

Visual effects of bit errors can be assessed in terms of pixels. Therefore, the error performance of the images was presented in a ratio called “Pixel Error Rate (PER)”, which indicates the degree of image degradation.

Through the analysis of the  $i^{\text{th}}$  received pixel’s variance from its transmitted value, a measure of visual disturbance,  $\Delta_i$ , can be quantified as in (1), where  $t_i$  and  $r_i$  represent the transmitted and received pixel colours respectively, for an  $n$  colour image.

$$\Delta_i = \frac{|r_i - t_i|}{n} . \quad (1)$$

From (1) it follows that the PER is calculated as in (2),

$$PER = \frac{1}{XY} \sum_{i=0}^{XY} \Delta_i . \quad (2)$$

where  $X$  and  $Y$  are the horizontal and the vertical resolution of the image, respectively.

As Fig. 3 shows, the performance of the Turbo-JPEG scheme is very poor in the 1.0 – 1.4 dB range. This is due to the inherent fragility of the JPEG structure and its inability to correct or restrict the propagation of any errors.

As expected, the performance of the BMP-turbo scheme is good throughout the range. This results from the complete independence of all pixels from one another. Hence, when errors cannot be repaired by the turbo decoder, only pixels with corrupted bits are affected.

Finally, as Fig. 3 clearly indicates, performance close to, and, as the channel improves, surpassing that of the BMP-turbo model is achieved by the turbo coded APEL. In the region after 1.175 dB, the post-processing techniques employed by APEL [2] recover many of the damaged pixels which could not be corrected in the case of BMP.

To observe the visual impact of data errors, samples of the various file formats have been decoded at a signal-to-noise ratio of 1.175 dB (Fig. 5-Fig. 7). To provide a qualitative reference for comparison, an uncoded version of the BMP file transmitted over the same channel has been included (Fig. 4).

In this example, although the PER is the same for images in Fig. 6 and 7, the subjective quality of the latter is slightly better. This results from the less frequent and clustered nature of the pixel errors in the APEL image, and the effects can be seen in more detail in the magnified areas in these figures.

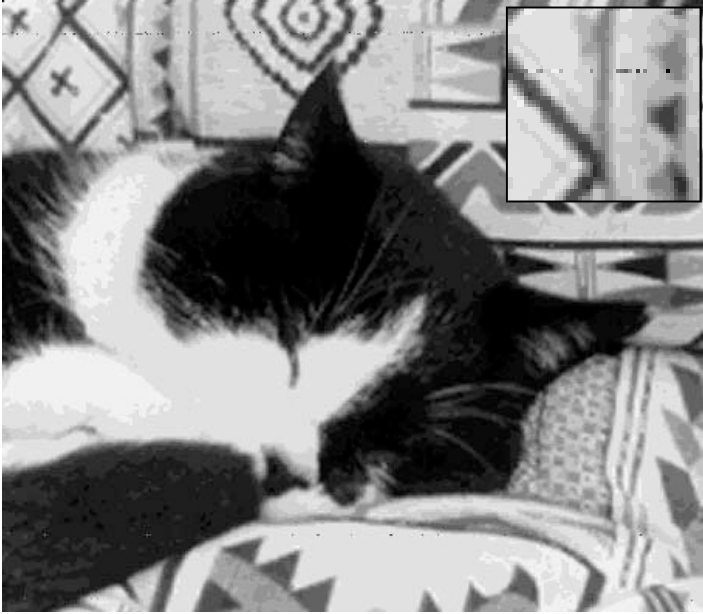
Since the APEL image coding scheme is lossless, the compression level relating to the JPEG format was reduced to provide a fair comparison, although even at this minimal level of compression the JPEG image was found to be greatly modified pixel-wise from that of the source. The attained compression ratio for these two formats, APEL and JPEG, was nevertheless still around 3 to 1, offering a substantial reduction over uncompressed BMP data.



**Fig. 4.** Uncoded BMP image transmitted through AWGN channel at 1.175 dB



**Fig. 5.** Turbo coded JPEG image transmitted through AWGN channel at 1.175 dB



**Fig. 6.** Turbo coded BMP image transmitted through AWGN channel at 1.175 dB



**Fig. 7.** Turbo coded APOL image transmitted through AWGN channel at 1.175 dB

Turbo coded JPEG, bitmap and APEL images were also transmitted over a Rayleigh channel with a maximum of 200 burst errors introduced randomly in each interleaver block. Figures 9 through 11 illustrate the system performance in the presence of burst errors. The uncoded BMP image has also been included to provide an insight into channel conditions (Fig. 8). Unlike the Gaussian channel, errors in the more severe Rayleigh case made for unreliable and inconsistent PER plots. It was observed that the dynamic range of the decoded image quality was wide in this case.

Due to the severe effects of burst errors, Turbo-JPEG fails to maintain data integrity and synchronisation after decoding (Fig. 9). The fragile structure of Huffman coding stage makes it almost impossible to withstand such channel conditions. In addition, since the turbo decoder is unable to correct burst errors, image transmission with Turbo-JPEG becomes very unreliable.

As illustrated in Fig. 8, channel errors are introduced as both randomly and in bursts. The Turbo-BMP scheme (Fig. 10) was observed to eliminate the majority of random errors effectively, however the majority of burst errors remained uncorrected. In other words, this scheme behaved similar to a 'burst-pass filter', where erroneous pixels appeared as trails of various lengths after decoding.

Turbo-APEL (Fig. 11) performance in the presence of burst errors, is visually comparable to that of Turbo-BMP (recall that the APEL image has 3:1 compression!). Channel errors which affect pels from various bit planes can be corrected through an analysis of the other planes. In other words, the post-processing techniques introduced by APEL coding, provide a powerful means of interpolating pixels using valid image information. Hence, the output of the 'burst-pass filter' can be further processed to correct more pixel errors than in the other cases. However, as the number of burst errors per information block is increased, distortion in APEL images, as anticipated, remain noticeable despite the post-processing.

Secondly, the interleaving stage in APEL coding, distributes pixel errors across the entire image (Fig. 11). Visually, small clustered errors are less disturbing to the eye than erroneous pixel trails (Fig. 10).

## Conclusions

We have proposed the combination of APEL and turbo coding in order to produce an enhanced image transmission system for low signal-to-noise ratios. Moreover, images in Fig. 6 and 10 require more bits to encode and transmit than images in Fig. 7 and Fig. 11; further underlining the advantages of the APEL turbo scheme outlined here.

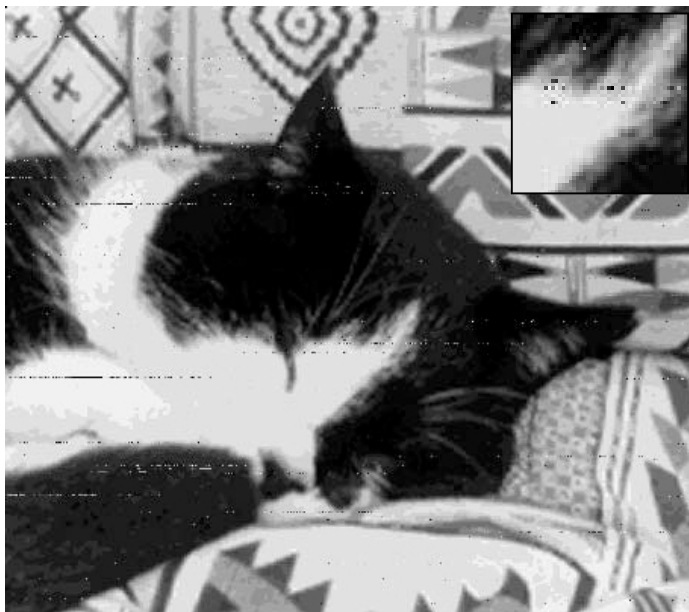
Even though APEL coding is used with a Turbo decoder that is not powerful enough to correct burst errors, the second interleaving stage in APEL is seen to minimise the visual impact of errors. This visual improvement is achieved by the spreading of burst errors across different bit planes, which provides an interleaver gain at the decoder.



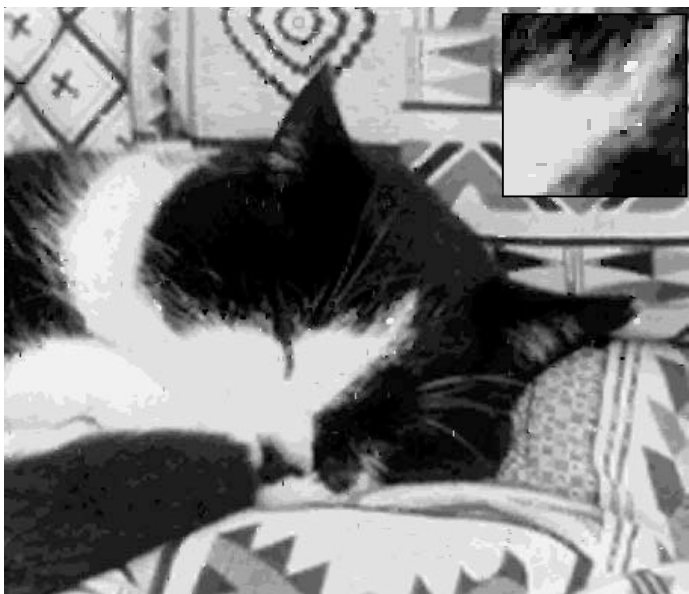
**Fig. 8.** Uncoded BMP image transmitted through Rayleigh channel at 4.0 dB



**Fig. 9.** Turbo coded JPEG image transmitted through Rayleigh channel at 4.0 dB



**Fig. 10.** Turbo coded BMP image transmitted through Rayleigh channel at 4.0 dB



**Fig. 11.** Turbo coded APEL image transmitted through Rayleigh channel at 4.0 dB

In addition, within the APEL image, whilst the majority of bit errors are corrected via iterative decoding, any which evaded detection (and thus perhaps falsely inserted as erroneous pixels) are restricted as a result of the robust data structure.

The novel combination of source/channel image coding technique, in this case APEL, with additional channel protection provides not only a resilience to Gaussian type errors, but also offers a powerful tool for the restriction and correction of burst errors.

To conclude, this approach can be further explored to develop integrated coding techniques, which could provide more reliable image communication means for noisy channels.

## Acknowledgements

The authors wish to thank DERA Malvern and NDS Ltd for their financial and technical support.

## References

1. Chippendale, P., Honary, B., Arthur, P. and Maundrell, M.: International Patent Ref.: PCT GB 98/01877, 'Data Encoding System', 1999
2. Chippendale, P.: 'Transmission of images over time-varying channels', PhD Thesis, August 1998
3. Berrou, C., Glavieux, A., Thitimajshima, P.: 'Near Shannon Limit Error Correcting Coding and Decoding: Turbo-Codes', IEEE Proc. ICC '93 Geneva, Switzerland, May 1993, pp. 1064-1070
4. Hagenauer, J., Offer, E., Papke, L.: 'Iterative Decoding of Binary Block and Convolutional Codes', IEEE Transactions on Information Theory, Vol. 42, No. 2, March 1996
5. International Organisation for Standardisation.: 'JPEG Digital Compression and Coding of Continuous-Tone Still Images'. Draft ISO 10918, 1991
6. NewScientist: 'The sky's the limit', No.2193, 03.07.1999, pp. 6

# Perfectly Secure Authorization and Passive Identification for an Error Tolerant Biometric System

George I. Davida<sup>1</sup> and Yair Frankel<sup>2</sup>

<sup>1</sup> Center for Cryptography, Computer, and Network Security,  
University of Wisconsin-Milwaukee, USA.

`davida@cs.uwm.edu`

<sup>2</sup> CertCo Inc., New York, NY, USA.

`yfrankel@cs.columbia.edu`, `yfrankel@cryptographers.com`

**Abstract.** A biometric identification system was recently developed and analyzed as a secure mechanism for user authentication. The system provided for the confidentiality, without the use of cryptographic encryption, of the user's biometric information stored in public verification templates.

Here we demonstrate that the use of majority decoding can enhance the prior techniques in several ways. One enhancement allows the biometric authentication system to leak no information about a user's biometric when using the proper computational assumptions. Another enhancement is a passive identification system.

## 1 Introduction

An Iris scan is a biometric technology which uses the human iris to authenticate users [BAW96, HMW90, Dau92, Wil96]. This technology produces a 2048 bit user biometric template such that any future scan of the same user's iris will generate a "similar" template. By similar, we mean having an acceptable Hamming distance within a predefined range, usually up to ten percent of the size of the code (e.g., Hamming distance between original reading and future reading is set to be in the range from 20 to 200). Moreover, the Hamming distance for the biometric readings of two different users has been shown to be much higher, about 45 percent (or 921 bits).

One can think of a biometric reading of a user as a faulty communication channel which may introduce a limited number of errors. Informally the typical biometric system works in the following manner. A user's biometric template is registered. A future reader compares the newly generated template with the registered template to test for closeness. With respect to iris scan technology closeness is measured by the Hamming Distance.

In the work [DFM98], the feasibility of protecting the privacy of a user's biometric and other security features were studied. It was suggested that providing additional privacy for the user's biometric may provide for stronger user acceptance. (For instance, an iris template may be used to determine some medical

conditions by an insurance company instead of the legitimate identification process the user submitted to.) The objective in [DFM98] was to allow protection of a user's biometric information in unprotected devices (such as a magnetic strip) or in a publicly accessible database (such as in a public key certificate). To address scalability concerns, private keys by the user or the reader was not used. Also, encryption is prone to loss of the cryptographic keys from the reader (i.e., the loss of a single key compromises every user).

Providing for authorization bound to a biometric template appears to be inherently difficult in this model, because the user's biometric template cannot exist in the clear on the storage device. To eliminate the need for storage of the biometric (in the clear or encrypted form), a new verification algorithm had to be developed.

The primary tools of the work of [DFM98] were error correcting codes (including majority decoding) and cryptographic methods. Later, the effectiveness of majority decoding was further analyzed by [DFMP]. Due to the effectiveness of majority decoding we are able to even further improve the error correcting codes (ECC)/Crypto based biometric system.

## 1.1 The Result

We enhance the system of [DFM98] in two ways:

- Though a biometric reading contains significant errors with respect to the original reading, we show how to use a biometric scan as an index in order to provide a scalable passive user identification system. By passive identification system we mean it can uniquely identify a user from a set of registered users through the use of only a biometric scan and no other input.
- We develop a system with perfect information preservation. The scheme in [DFM98] leaks approximately as many bits, in an information theoretical sense, as there are redundancy (error correcting) bits to correct errors in a biometric template. However, there was no way to quantify what kinds of bits are leaked and under what assumptions. Here we demonstrate that in a computational model, majority decoding can be used in a manner that no information is leaked with sufficiently high probability.

## 2 Background

### 2.1 Error Correcting Codes

**Majority decoding:** In the rest of the paper, we will consider only binary error correcting codes. We will denote by  $a \parallel b$  the concatenation of two strings  $a, b$ .

Let  $\mathbf{v}_i = \langle v_{i,1}, v_{i,2}, \dots, v_{i,n} \rangle$  be  $n$  bit code vectors. Given odd  $M$  vectors  $\mathbf{v}_i$ , a majority decoder computes vector  $\mathbf{V} = \langle V_1, V_2, \dots, V_n \rangle$ , where

$$V_j = \langle \text{majority}(v_{1,j}, \dots, v_{M,j}) \rangle,$$

i.e.,  $V_j$  is the majority of 0's or 1's of bit  $j$  from each of the  $M$  vectors. We shall use majority decoding primarily to get the best biometric reading possible, thus reducing the Hamming distance between successive *final* readings  $\mathbf{V}$ .

In the biometric authentication protocol, described in Section 2.2 the biometric being measured will be estimated by sampling since the actual unique iris is not measured with precision. The samples that are taken of the iris will converge to the actual unique individual biometric, with majority decoding, with high probability.

**Error correction:** An  $[n, k, d]$  code [Ber68, MS78, PW88] is a code of  $n$  bit codewords (vectors) where  $k$  is the number of information digits and  $d$  is the minimum distance of code. Such a code can correct at least  $t = (d - 1)/2$  errors.

**Note: Bounded distance decoding:** In the rest of the paper, we assume that the decoding performed at the point of verification is to correct at most  $(d - 1)/2$  errors. This is necessary to ensure that no bogus biometric is decoded into a valid one. Bounded distance decoding can be readily implemented through a simple count of the Hamming weight of the error vector computed. In some decoding schemes, the error locations that are computed are the roots of some polynomial  $\sigma(z)$  over  $GF(2^m)$  of degree  $t = \text{degree}(\sigma(z))$ . If  $t > t = (d - 1)/2$  then the biometric is rejected.

## 2.2 An Error Correcting Based Biometric System

The primary observation of [DFM98] is that a user's biometric template can be viewed as the information bits of an error correcting code. Now instead of storing the biometric template only the error correction bits are necessary on the storage device, a magnetic strip card for simplicity<sup>1</sup>. Since only the check bits are stored on the user's card, the available information about the biometric template is reduced. On the other hand, the reader can take a new reading of the user's biometric template, append the ECC check bits, remove the errors using bounded distance decoding, and finally, with high probability, reproduce the original template, which can be verified with the signature on the token.

One other hurdle has to be overcome to provide security. The signature may itself leak the user's template. Observe that  $\langle M, \text{SIG}(M) \rangle$ —is a signature for message  $M$  which leaks all bits of  $M$ , yet is a valid signature of  $M$ . To resolve this problem, special hash functions were used in [DFM98].

Here is a brief summary of the basic off-line biometric protocol presented in [DFM98].

*System Setup:* The authorization center generates its public and private keys and disseminates its public key to the biometric readers. The system also sets up an  $[n, k, d]$  code.

*User Initialization:* To register,  $M$  biometric templates of length  $k$  are independently generated for the legitimate user. Majority decoding is then applied to

<sup>1</sup> A smartcard, a database record, a public certificate can be stored as well.

the  $M$  biometrics to obtain the user's  $k$  bit template  $\mathbf{T}$ . Given the  $k$  information digits  $\mathbf{T}$ , an  $n$  digit codeword  $\mathbf{T}-\mathbf{C}$  is constructed, where  $\mathbf{C}$  are the check digits, in the  $[n, k, d]$  code defined at system setup. A storage device is constructed with the following information:

1. Name of the individual, NAME.
2. Other public attributes ATT, such as the issuing center and a user's access control list.
3. The check digits  $\mathbf{C}$ , of the biometric.
4.  $\text{Sig}(\text{Hash}(\text{NAME}, \text{ATT}, \mathbf{T}-\mathbf{C}))$  where  $\text{Sig}(x)$  denotes the authorization officer's signature of  $x$ , and  $\text{Hash}(\rightarrow)$  is a partial information hiding hash function [Can97] (e.g.,  $\text{Sig}(\text{Hash}(\rightarrow))$  is a content-hiding signature) or a random oracle (See [BR93]).

*Biometric verification process:* When a user presents herself/himself and the card with the information described above,  $M$  biometric templates are independently generated for the user. Majority decoding is applied to the  $M$  biometric vectors to obtain the user's  $k$  bit template  $\mathbf{T}$ . Error correction is performed on codeword  $\mathbf{T}-\mathbf{C}$  to obtain the corrected biometric  $\mathbf{T}$ . The signature  $\text{Sig}(\text{Hash}(\text{NAME}, \text{ATT}, \mathbf{T}-\mathbf{C}))$  is then verified. Successful signature verification implies the user passed the identification step. For simplicity of exposition, we assume that occasional rejection of a valid user is acceptable (the user would simply repeat the scan). In applications where rejection of a valid user is not acceptable, the parameters of the system can be changed so that such an event has negligible probability. Determining the correct parameters in such a case involves bounding the area under the tail of a binomial distribution (or a Normal approximation to the binomial distribution via the Central Limit Theorem).

Proof of security and in particular the choice of hash functions were discussed in [DFM98]. Moreover, the usefulness of majority decoding to detect impostors was discussed in [DFMP]

### 3 New Techniques

We now discuss the two new techniques: perfect confidentiality and passive identification. The new techniques are based on the following observation: Given a 2048 bit iris code, majority decoding is used on a sufficient number of samples to reduce the expected number of errors to a small number, e.g. 1 per block of 2048 bits.

No. of scans	Per bit prob. of error	Expected no. of errors in a 2Kb scan
1	0.1	205
3	0.028	58
11	0.000306	1
21	0.00000135	.002

Once a reduced-error iris code is obtained using majority decoding, we construct  $D$  indices  $I_j, 1 - j - D$  from I-SIZE subsets of  $\mathbf{T}$  with the GEN-Index function as follows:

**Step 1** Set  $j = 0$

**Step 2** Set  $j = j + 1$

let  $X_j = \text{PermutedChoice}(j, \mathbf{T})$  be I-SIZE bits of the biometric  $\mathbf{T}$  selected with schedule  $PC$ , where I-SIZE is chosen so that the entropy of the  $X_j$  bits is sufficient( e.g. For an iris scan as described above if I-SIZE=600 bits, then, assuming that the biometric has an entropy of 160, the entropy of  $X_j$ , on average, satisfies  $H(X_j) = 53$ ).

let  $I_j = \text{hash}(X_j)$

**Step 3** if  $(j < D)$  goto 2 else exit.

These indices  $I_j$  are pointers to the database locations where the user templates are stored. Collisions with other iris codes is dealt with by performing the checks to be described later.

Two important observations can be made. First, with high probability at least one index is error free (See Majority Decoding in section 2.1). Second when  $\text{hash}(\rightarrow)$  has same the information hiding property as those used in [DFM98] (see [Can97] as an example) and  $X_j$  has sufficient entropy, the  $I_j$  leak no useful information about the iris.

### 3.1 Passive Identification

In a passive identification system the user is uniquely identified with only the biometric reading and without any other inputs from the user. Hence the user does not provide an ID number or other inputs via a keyboard or a smart card. Once the user's biometric is read, the user must be uniquely identified to obtain a user id. This may be done by a linear search through a database of registered biometric/user attributes relationship database. However, in practice a linear search is not scalable for applications with a large user base.

In a biometric system, such as iris scan, there exists variances from the original registered reading with a later acquired reading. Because of the variances, it is not possible, in general, to use biometric systems as a scalable passive identification systems. Scalability becomes difficult because if the reading is faulty and lacking any other input from the user due to the passive nature of the identification scheme, the biometric can no longer be an index into a registered template database and therefore only linear searches are generally possible.

As discussed above, we note that using majority decoding with iris scan technology one is able to reduce the number of errors to a negligible amount. This is based on observations that the errors in successive readings of a biometric differ in positions that are randomly distributed over the iris code, with about 10 percent Hamming distance between success readings, on the average. Assuming

that the errors are random over the code: they can be reduced through majority decoding of  $M$  independently read iris code vectors.

Let  $\mathbf{T}$  be the template for an individual who presents to the authorization center. For each such user, we construct  $D$  indices  $I_j, 1 \leq j \leq D$ , of size I-SIZE as described above, which are pointers to the location of the record. Standard hashing techniques can be used to produce the indices.

We now define the following identification system:

To register,  $M$  biometric templates of length  $k$  are independently generated for the legitimate user. Majority decoding is then applied to the  $M$  biometrics to obtain the user's  $k$  bit template  $\mathbf{T}$ . Given the  $k$  information digits  $\mathbf{T}$ , an  $n$  digit codeword  $\mathbf{T}-\mathbf{C}$  is constructed, where  $\mathbf{C}$  are the check digits, in the  $[n, k, d]$  code defined at system setup. In the secure database the following information is stored:

1. Name of the individual, NAME.
2. Other public attributes ATT, such as the issuing center and a user's access control list.
3. The check digits  $\mathbf{C}$ , of the biometric.
4. Hash(NAME, ATT,  $\mathbf{T}-\mathbf{C}$ ) where Hash( $\rightarrow$ ) is a partial information hiding hash function [Can97]<sup>2</sup>.

The database is set up so that the indices  $I_1, \dots, I_D$  created from  $\mathbf{T}$  with the GEN-Index function link to the created record.

*Passive identification process:* During verification, when a user presents herself/himself, the verification unit performs the following steps

- Step 1** Set  $i = 0$ ,  $M$  biometric templates are independently generated for the user. Majority decoding is applied to the  $M$  biometric vectors to obtain the user's  $k$  bit template  $\mathbf{T}$ .
- Step 2** Set  $i = i + 1$ , Construct index  $I_i$  with the GEN-Index function on input  $\mathbf{T}$ .
- Step 3** The records pointed to by indices  $I_i$ , containing the check digits and hash value, are requested. Let  $\mathbf{C}_i$  be the check digit in record indexed by  $I_i$ . Each set of check digits  $\mathbf{C}_i$  is then used along with  $\mathbf{T}$  to produce a new corrected biometric  $\mathbf{T}_i$ .
- Step 4** The hash value Hash(NAME, ATT,  $\mathbf{T}_i - \mathbf{C}$ ) is then compared for equality with the hash value received.
- Step 5** If success, exit (success)
- Step 6** If  $i < D$  go to 2 else exit(failure)

Successful verification implies the user passed the identification step. The NAME and the ATT fields identify the user uniquely. Observe that with overwhelming probability at least one of the indices will be correct. In fact, there

<sup>2</sup> A signature, private key authentication, hash, etc. can be used as well depending on the security model.

will most likely be multiple indices pointing to the same record. To reduce the number of queries into the database those records pointed by the most indices should be tested first.

### 3.2 Perfect Secrecy

User acceptance is vital for any biometric system to be effective. However, most systems reveal information about the user in the registration template. Systems based on the iris measurements may be particularly sensitive to revealing health information in the template.

In [DFM98, DFMP], the protocol presented leaks only as much information as the error checking bits included in the template. However, this is an information theoretical analysis and it does not say anything about information leakage in a computational sense.

What we desire is an identification system that achieves perfect secrecy, without storing the biometric. Informally, perfect secrecy means that an polynomial time adversary given a registration template is unable to compute any information about the user biometric related to the template.

Here we use a technique very similar to the passive identification. Define  $PRF(.,.)$  to be a Pseudo Random Function with two inputs. We obtain  $D$  indices as before but this time we store on the user token  $D$  tuples  $\langle R_j, C_j \rangle$  for  $1 \leq j \leq D$ , where  $C_j = PRF_{I_j}(R_j) - C$  and  $R_j$  is a random string. This in essence encrypts  $C$  under each of the keys  $I_j$ .

*System Setup:* The authorization center generates its public and private keys and disseminates its public key to the biometric readers. The system also sets up an  $[n, k, d]$  code.

*User Initialization:* To register,  $M$  biometric templates of length  $k$  are independently generated for the legitimate user. Majority decoding is then applied to the  $M$  biometrics to obtain the user's  $k$  bit template  $T$ . Given the  $k$  information digits  $T$ , an  $n$  digit codeword  $T-C$  is constructed, where  $C$  are the check digits, in the  $[n, k, d]$  code defined at system setup. Let  $I_1, \dots, I_D$  be the  $D$  indices chosen as described above. A record (stored on a token to be carried by the user) is constructed with the following information:

1. Name of the individual, NAME.
2. Other public attributes ATT, such as the issuing center and a user's access control list.
3.  $\langle R_j, C_j \rangle$ ,  $1 \leq j \leq D$ , where  $C_j = PRF_{I_j}(R_j) - C$ ,  $I_j$  are  $D$  indices of size I-SIZE and  $R_j$  is a random string.
4.  $Sig_j = Sig(\text{Hash}(\text{NAME}, \text{ATT}, T-C_j))$ ,  $1 \leq j \leq D$ , where  $Sig(x)$  denotes the authorization officer's signature of  $x$ , and  $\text{Hash}(\rightarrow)$  is a partial information hiding hash function [Can97] (e.g.,  $Sig(\text{Hash}(\rightarrow))$  is a content-hiding signature) or a random oracle (See [BR93]).

*Biometric verification process:* When a user presents herself/himself and the card with the information described above, the following steps are performed

**Step 1** set  $j = 0$

$M$  biometric templates are independently generated for the user. Majority decoding is applied to the  $M$  biometric vectors to obtain the user's  $k$  bit template  $\mathbf{T}$ .

**Step 2**  $j = j + 1$

Compute  $I_j$  with the GEN-Index function on input  $\mathbf{T}$ .

Compute  $\mathbf{C}_j = \text{PRF}_{I'_j}(R_j) - \mathbf{C}_j$ .

Apply error correction on codeword  $\mathbf{T} - \mathbf{C}_j$  to obtain the corrected biometric  $\mathbf{T}_j$ .

**Step 3** The signature<sup>2</sup>  $\text{Sig}_j = \text{Sig}(\text{Hash}(\text{NAME}, \text{ATT}, \mathbf{T}_j - \mathbf{C}_j))$  is then checked. A successful signature verification implies the user passed the identification step.  
exit(success)

**Step 4** If  $i < D$  go to 2 else exit(failure)

Informally, the reasons this scheme attains perfect secrecy are: Observe that  $\langle C_1, R_1 \rangle, \dots, \langle C_D, R_D \rangle$  are multiple encryptions each of  $\mathbf{C}$  with a key (index) with sufficient entropy. That is each key has around 53 bits entropy, as discussed above, but more can be added. Now each of the keys (indices)  $I_j$  operates on a random  $R_j$  to provide independence amongst the tuples. If a random oracle rather than pseudo-random function is used then the random values  $R_j$  are not necessarily needed.

### 3.3 Passive Identification with Untrusted Verifier

In the passive identification protocol above the reader performed the final verification process. That is it verified the signature. If it is desired that this verification step be performed by the central database holder, without leaking information about the user's biometric, then using a random oracle model we can solve this problem by combining the presented techniques.

As in the passive identification, indices are generated and a user's information is stored in a manner which allows the indices to point to the appropriate data. However, this time the user information is different:

*System Setup:* The authorization center generates its public and private keys and disseminates its public key to the biometric readers. The system also sets up an  $[n, k, d]$  code.

*User Initialization:* To register,  $M$  biometric templates of length  $k$  are independently generated for the legitimate user. Majority decoding is then applied to the  $M$  biometrics to obtain the user's  $k$  bit template  $\mathbf{T}$ . As in the GEN-Index function, let  $X_j = \text{PermutedChoice}(j, \mathbf{T})$  be the I-SIZE random bits of the vector  $\mathbf{T}$ . Now, for  $\text{RO}(\cdot)$ , a random oracle (see [BR93]), let  $\mathbf{T}_j = \mathbf{T} - \text{RO}("0" - X_j)$

<sup>2</sup> Hashes, private key authentication etc. can be used instead depending on the security model.

and  $I_j = \text{RO}("1" - X_j)$ . Given the  $k$  information digits  $T_j$ , an  $n$  digit codeword  $T_j - C_j$  is constructed, where  $C_j$  are the check digits, in the  $[n, k, d]$  code defined during setup. In the database we store at a location pointed to by indices  $I_j$ :

1. Name of the individual, NAME.
2. Other public attributes ATT, such as the issuing center and a user's access control list.
3. The check digits of the encrypted biometric:  $C_j, 1 - j - D$ .
4.  $D$  hashes  $\text{Hash}(\text{NAME}, \text{ATT}, T - C_j), 1 - j - D$ , where  $\text{Hash}(\rightarrow)$  is a partial information hiding hash function [Can97]<sup>3</sup>.

*Biometric verification process:* When a user presents herself/himself,  $M$  biometric templates are independently generated for the user. Majority decoding is applied to the  $M$  biometric vectors to obtain the user's  $k$  bit template  $T$ . As in the GEN-Index function, let  $X_j$  be the I-SIZE bits of  $T'$  selected using schedule  $PC$ , as described above. The reader sends to the database server tuples  $\langle I_j, T_j \rangle$ —where  $T_j = T - \text{RO}("0" - X_j)$  and  $I_j = \text{RO}("1" - X_j)$ . The server finds the user's records from the  $I_j$ . Error correction is performed, for each  $i$ , on codeword  $T_i - C_i$  to obtain the corrected biometric  $T_i$  by the database server. The hashes  $\text{Hash}(\text{NAME}, \text{ATT}, T_i - C_i)$  are then checked. Successful verification implies the user passed the identification step. For simplicity of exposition, we assume that occasional rejection of a valid user is acceptable (the user would simply repeat the scan). In applications where rejection of a valid user is not acceptable, the parameters of the system can be changed so that such an event has negligible probability. The reader is then informed of the success or failure of the verification by the central server.

Observe that the  $C_j$  leak no information because all possible  $T_j$  are equally likely given that  $\text{RO}(\rightarrow)$  is a random oracle. For correctness, observe that for valid user  $u$  with subsequent reading  $T$  has an error vector  $E = T - T$ . Suppose  $X_j$  is the "index" without any errors. Then performing error correction on  $T - C_j = T - E - \text{RO}("0" - X_j) - C_j$  returns  $T_j - C_j$  because  $E$  has low Hamming weight. Also note that we use two different random oracles  $\text{RO}("1", \rightarrow)$ , for the indices, and  $\text{RO}("0", \rightarrow)$ , for the keys to encrypt a users template. This allows us to use the same bits of the template in two ways without leaking the key (i.e.,  $\text{RO}("0" - X_j)$ ) for a key and index  $\text{RO}("1" - X_j)$ .

**Computationally Simple Passive Identification:** Using the same idea as described above a computationally simpler and heuristically secure mechanism can be constructed. At the setup process vectors  $T_j = T - \text{RO}("0" - X_j)$  and  $I_j = \text{RO}("1" - X_j), 1 - j - D$ , are stored. In the biometric verification process  $T$  is obtained as before. Vectors  $T_j = T' - \text{RO}("0" - X_j)$  and  $I_j = \text{RO}("1" - X_j)$ , where  $X_j$  is created from  $T$  as before, using a Permuted Choice schedule, are now created. Acceptance occurs when there exists a  $T_j$  whose Hamming weight is sufficiently close to the retrieved vector  $T_j$ , retrieved with  $I_j$ . Observe in all the perfect security schemes no additional information is leaked if different

<sup>3</sup> A signature can be used as well.

authorization centers uses different parameters (e.g., PermutedChoice,  $[n, d, k]$  code, random oracle, etc.).

In the non-passive case when the user is allowed to provide some information to the reader (e.g., a magnetic strip card containing error correction bits for the user's template), to provide for an untrusted verifier then as in [DFM98] a hashed biometric template regenerated by the reader can then be used as a key. This key can be used as an authentication key for a challenge response where challenge is generated by the untrusted verifier. That is, let  $K = \text{RO}(\mathbf{T})$  be stored by the untrusted verifier. Verification is response  $f_K(C)$  where  $C$  is a challenge from untrusted verifier or generated by a random oracle, at the reader, with some one-time tag (i.e., using inputs such as time, date, random values, names, etc.).

## References

- [BAW96] F. Bouchier, J. S. Ahrens, and G. Wells. Laboratory evaluation of the iris-can prototype biometric identifier. Technical Report SAND96-1033, Sandia National Laboratories USA, April 1996.
- [Ber68] E. R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, 1968.
- [BR93] M. Bellare and R. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computers and Communications Security*, 1993.
- [Can97] R. Canetti. Towards realizing random oracles: Hash functions which hide all partial information. In *Advances in Cryptology. Proc. of Crypto'97*, pages 455–469, 1997.
- [Dau92] J. Daugman. High confidence personal identifications by rapid video analysis of iris texture. In *IEEE International Carnahan Conference on Security Technology*, pages 50–60, 1992.
- [Dau93] J. Daugman. High confidence personal identifications by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):648–656, November 1993.
- [DFM98] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *1998 IEEE Symposium on Security and Privacy*, pages 148–157, 1998.
- [DFMP] G. I. Davida, Y. Frankel, B. Matt and R. Peralta, ' On the relation of error correction and cryptography to an offline biometric based identification scheme. In *Proceedings of the Workshop on Codes and Cryptography 1999*.
- [HMW90] J. P. Holmes, R. L. Maxell, and L. J. Wright. A performance evaluation of biometric identification devices. Technical report, Sandia National Laboratories, July 1990.
- [MS78] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. North – Holland Publishing Company, 1978.
- [PW88] W. W. Peterson and E. J. Weldon. *Error Correcting Codes*. The MIT Press, 1988.
- [Wil96] G. O. Williams. Iris recognition technology. In *IEEE International Carnahan Conference on Security Technology*, pages 46–59, 1996.

# An Encoding Scheme for Dual Level Access to Broadcasting Networks

Thumrongrat Amornraksa, David R.B. Burgess, and Peter Sweeney

CCSR, University of Surrey, Guildford, GU2 5XH, U.K.

t.amornraksa, d.burgess, p.sweeney@ee.surrey.ac.uk

**Abstract.** In this paper, we propose an encoding scheme which gives two levels of access to a broadcast encrypted signal. Watermarking-type techniques based on direct-sequence spread spectrum communications are implemented to add specific information to the signal within the bandwidth allocated for broadcasting. This is beneficial to both the service providers and all subscribers in the network since the information added can advertise programmes which many are not yet authorised to access.

## 1 Introduction

An advantage of communications over the broadcasting network is that the transmitted signal from a source station can be received simultaneously by many destination stations. Digital TV broadcasting is one of the applications that uses this advantage. Since some digital TV programmes are pay-TV services, they will be encrypted before transmitting to every subscriber. Only the authorised subscribers who pay an extra fee can get access to those programmes. This technique does not give any value at all to other subscribers who have not paid for that particular programme. The allocated bandwidth is only used for broadcasting the encrypted signal to the authorised subscribers, which may be a small group compared to all subscribers in the network. It will be more efficient if we can devise an encoding scheme in which the authorised subscribers can access the encrypted signal and, at the same time, the other subscribers can receive something on the same channel, such as an advertisement. However, the scheme should not extend the existing allocated bandwidth.

In this paper, we propose such an encoding scheme which gives two levels of access to the subscribers in the network. Watermarking-type techniques based on direct-sequence spread spectrum communications are implemented to add specific information (i.e. advertisements) to the access-limited signal, which is protected by encryption techniques. With this scheme, the allocated bandwidth for broadcasting is utilised more efficiently and more benefit is given to both the service providers (through advertising) and all subscribers in the network (since there will be programmes which they are not authorised to access but can see advertised).

## 2 Description of the Scheme

In spread spectrum (SS) communications [1], a low level wideband signal can easily be hidden within the same spectrum as a high power signal where each signal appears to be noise to the other. The heart of these SS systems is a pseudo-random binary sequence (PRBS). For these direct sequence SS systems, the original baseband bit stream is multiplied by the PRBS to produce a new bit stream. Only those receivers equipped with the correct PRBS can decode the original message. At the receiver, the low level wideband signal will be accompanied by noise, and by using a suitable detector/demodulator with the correct PRBS, this signal can be squeezed back into the original narrow baseband. Because noise is completely random and uncorrelated, the wanted signal can easily be extracted.

Several watermarking techniques, such as those proposed in [2, 3], are based on these ideas. By spreading the information bits and modulating them with a PRBS, the watermark signal can be obtained. This signal is then embedded in the video signal below the threshold of perception. The recovery of the embedded watermark signal can be accomplished by correlating the watermarked video signal with the same PRBS that was used in the process of constructing the watermark signal. Correlation here is demodulation followed by summation over the width of the chip-rate (the number of blocks over which each information bit is spread). If the peak of the correlation is positive (or, respectively, negative), the recovered information bit is a +1 (or -1).

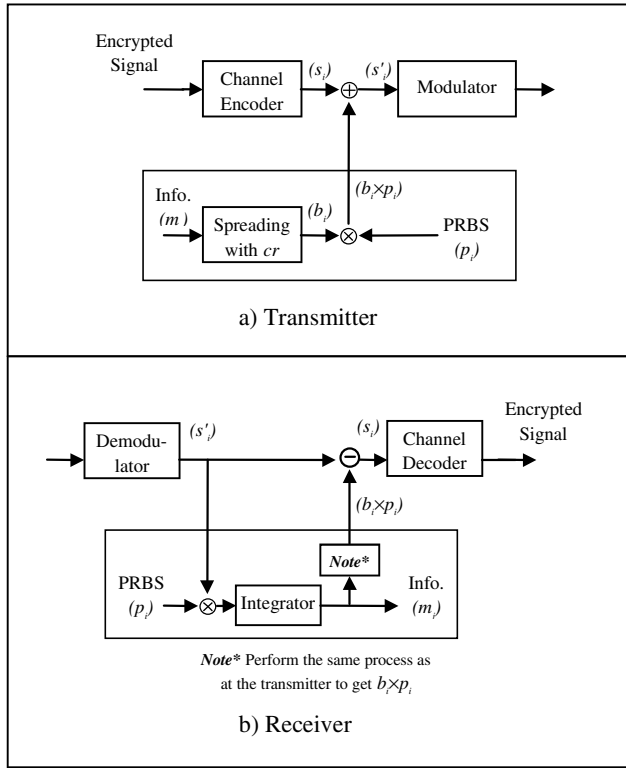
Using a similar technique to the above (particularly like that proposed in [3]), the information signal will be added to the encrypted signal (after the channel coding process) to give the signal for transmission. Given a key to reproduce the same PRBS at the receiver's side, the information signal can be recovered. Then the encrypted signal can be recovered by subtracting the information signal from the transmitted signal. Any errors which occur at this stage (both communication channel errors and any resulting from the need to ensure that the signal for transmission uses the same block size for its symbols as does the encrypted signal) will be detected and corrected by the channel decoder. The operation of the encoding scheme is shown in Figure 1 below.

We now describe the basic steps of adding the information signal to the encrypted signal. We denote by  $(m_j)$ ,  $m_j \in \{-1, 1\}$  a sequence of information bits we want to add to the encrypted signal. This discrete signal is spread by a large factor  $cr$ , the chip-rate, to obtain the spread sequence  $(b_i)$

$$b_i = m_j, j.cr \leq i < (j+1).cr \quad (1)$$

The spread sequence  $(b_i)$  is then modulated with a PRBS  $(p_i)$ ,  $p_i \in \{-1, 1\}$  and added to the encrypted signal  $s_i$ , each  $s_i$  block containing  $k$  bits, yielding the following signal for the modulation process:

$$s'_i = s_i + p_i.b_i \quad (2)$$



**Fig. 1.** The operation of the encoding scheme

At the receiver, the recovery of the added information is easily accomplished by multiplying the transmitted signal with the same PRBS ( $p_i$ ) that was used in the encoder. The summation over the correlation window i.e.  $cr$  is as follows:

$$r_j = \sum_{i=j.cr}^{(j+1).cr-1} p_i \cdot s'_i = \sum_{i=j.cr}^{(j+1).cr-1} p_i \cdot s_i + \sum_{i=j.cr}^{(j+1).cr-1} p_i^2 \cdot b_i \quad (3)$$

The first term on the right-hand side of (3) vanishes if  $p_i$  and  $s_i$  are uncorrelated and  $\sum_{i=j.cr}^{(j+1).cr-1} p_i = 0$ . However, we account for a different number of -1's and 1's in  $p_i$  over the interval  $[j.cr, (j+1).cr-1]$  by including the term

$$\Delta = \left( \sum_{i=j.cr}^{(j+1).cr-1} p_i \right) \cdot (s'_i) \quad (4)$$

Then  $r_j$  ideally becomes

$$r'_j = \sum_{i=j.cr}^{(j+1).cr-1} p_i \cdot s'_i - \Delta \approx cr \cdot m_j \quad (5)$$

and the recovered information bit  $m'_j = \text{sign}(r'_j)$ .

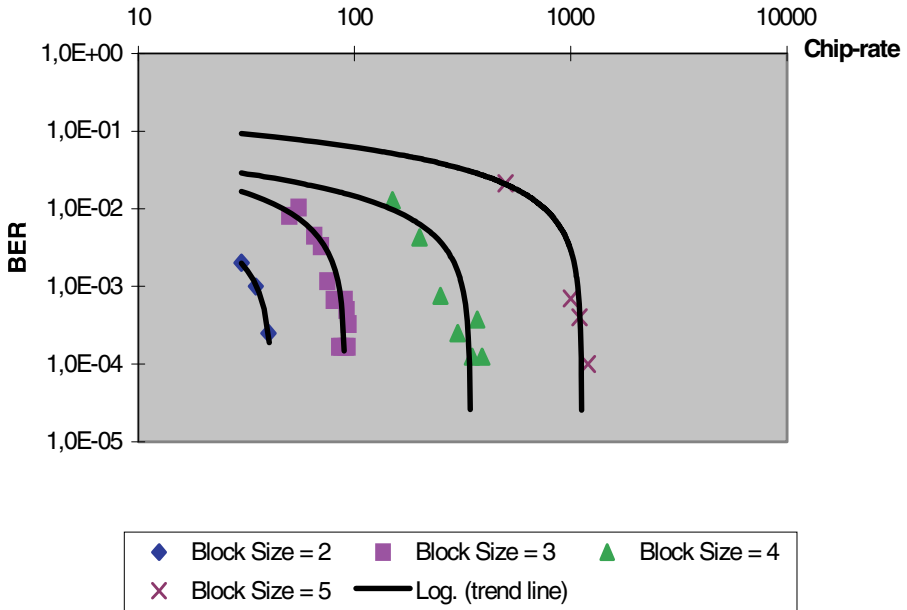
As an example, let the bit-rate of the encrypted signal be 3Mb/s, the chip-rate  $cr = 500$  and let the block size  $k$  be 4 bits. Then, the rate at which information bits can be added after the channel coding process is 1.5kb/s. With this bit-rate, the information signal can be an image signal in compression form transmitted every 30s or so, and we can transmit the total bit-rate of 3.0015Mb/s within the existing bandwidth allocation of 3Mb/s. To increase the bit-rate of the information signal, the chip-rate and the block size should be reduced. However, a smaller block size implies a greater likelihood that subtracting the information signal from the transmitted signal will not give the encrypted signal. In addition, a smaller chip-rate implies a greater likelihood of error in decoding the information bits. To reduce this latter likelihood of error an error correcting code can be applied to the information bits before the spreading process.

### 3 Experimental Results

Experiments were carried out using the programming language C. The block size was varied from 2-5 bits to represent up to 32 values. In the experiments, the smallest chip-rate which gave no errors after the decoding process was 41, 95, 470, 1300 for a block size of 2, 3, 4, 5 respectively. For these block sizes, other values of the chip-rate considered gave different values of Bit Error Rate (BER) in the decoded data, and these values and the underlying trend line are shown in Figure 2.

From Figure 2, it can be seen that a larger block size needs a higher chip-rate to retain the same BER. In addition, since one single bit error in the decoded information signal causes error propagation in the encrypted signal, anything other than a large value of the chip-rate will result in a large value of BER. This means that implementing the scheme with a large block size, will lead to low efficiency. According to the experimental results, a block size of 3 is the optimum for the scheme.

In our experiments, the encoding scheme was performed in an error-free communication channel. That is, the errors which occurred in the decoded data came solely from the need to remain within the bandwidth of the broadcast channel. Although the proposed scheme has not been fully explored, it shows an idea of how to utilise the existing broadcast bandwidth in a more efficient way. Further work can be carried out by simulating the scheme in channel models e.g. AWGN. Error correcting codes can be applied in the scheme to improve its reliability.



**Fig. 2.** Chip-rate vs. bit error rate of decoded data at different block sizes

## References

1. Pickholtz, R., Schilling, D. and Millstein, L.: Theory of Spread Spectrum Communications. A Tutorial. IEEE Transaction on Communication, Vol. COMM-30 (1982) 855-884
2. Cox, I., Kilian, J., Leighton, T. and Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia. IEEE transactions on Image Processing, Vol. 6, No. 12 (1997) 1673-1687
3. Hartung, F. and B. Girod, B.: Watermarking of Uncompressed and Compressed Video. Signal Processing, Vol. 66, no. 3 (Special issue on Watermarking) (1998) 283-301

# Photograph Signatures for the Protection of Identification Documents

no<sup>1</sup> o n on<sup>2</sup> n<sup>3</sup>

<sup>1</sup> ENIB, Brest, France

<sup>2</sup> Department of Electrical & Electronic Engineering,  
University of Wales Swansea, SA2 8PP, UK

<sup>3</sup> Dynjab Technologies, Canberra, Australia

J.S.D.Mason@swansea.ac.uk

**Abstract.** This paper investigates a photo-signature approach to protecting personal identification documents such as passports. The approach is based on that described in a recent publication by O’Gorman and Rabinovich [1] which uses encoded data derived from comparisons of image sub-blocks across the photograph of the document. The encoded data is generated and stored at the time of document creation, and used subsequently to test document authenticity. Here we report on experiments which corroborate the fundamental findings of [1], namely that it is possible to usefully encode the photograph information in only tens of bytes of data.

Furthermore we show that new block structures can improve the efficiency of the encoded data. This is important since the encoding efficiency, measured in terms of number of bytes versus discriminating performance, is particularly important when storing data on a small document such as a passport or ID card. We show that a step structure is measurably better than the original octal structure used in [1] when there are only a small number of bytes (20 to 30) in the photo-signature.

## 1 Introduction

o o p n on o n p po n  
p p n n n on o on o p on n p on n  
oo p n n on p po n p n n  
p po n n n on o on o p on n n o no  
o n p n o p n on  
– p n o n o p po  
– p po no p o n  
– p po n n o n n n  
– p po n o n o n no n p n  
n nn  
o on o n on on n  
p p n p o o p n o n p o p n n  
o o p po n n p po p

n on p o o p n n o n  
o n o n o o o n n n p o o p  
o on on n o n n p n o o n o  
o p n n op p o on o n n  
n n nn n o o o p n n o n no o  
n on n n o n n n o o  
p n n o o o o p o n o  
p n p n on o p o o  
p on o p n n  
on o n o p o o p o  
o n o o n on n  
n o p o o p *photo-signature* o o on  
n o n o on n  
n p o p p n o  
p p on p o o n n o n  
n no o o n on  
o on n n o n n n o p n  
on o n o n n n on  
p o n n n o n o o

2 Photo-Signatures against Counterfeiting

p o o n n n n p on o p o o p p o  
n n n on n p o o p o n p o  
p p opo pp o o n n n p op o  
p o o p n o o n n  
o o o p p o o p on p n p  
p op on n o p o n p o o p n  
o o o n o on n  
o n n p o on o  
o p o o n o on po n n n  
o n p o o p o n n  
o nn n o o p o n n o n n o n  
p o o n o pon n o p o o p n o  
on on o n n on no n o no  
n o n o n on n o o o n n  
pp op o o n n o n  
p pp n p o o p

2.1 Passport Protection

o p po n n n on n o o o  
p po on o n n on o on o

n n on o o n p po n n o on  
 o n n on o o n  
 p po n n o p  
 o p n n on o n o o n o p o o  
 n o po n o p po o on o n o on  
 n o n n n o o p po o o n  
 o on o n n p po o o n on  
 n o o no n o p  
 o n o n n n o n  
 o n n p o oo o p  
 n o o o o o on o p p  
 n o n o o o po o o n o n  
 o n o n o n n  
 n n p o o p p  
 n on o on p o o n o o  
 pp op o p po n o p o o p on p  
 o n n o n n p n o p po n  
 o p opo o n n no o  
 n p n o p po o o n o  
 p o o n po n o on n n  
 n n o n n o o  
 on o p o o p p n p po p o o p o n  
 n o p o o n o on o n n o  
 o pon n o on no o n o n  
 o o n n n oo o  
 n o n no n o n n o  
 p po n o o n o  
 o o n o o n o p o n p n n  
 p o o n p o o p n on p  
 p o p o

## 2.2 The Photo-Signature of [1]

o p o o n o po n p n p o  
 p o o p o o n o on n n o  
 o n o o n n on n p opo p o o n o  
 o n n no on p p n n o  
 o p po o p po p o on n on o  
 p o o n o n o n n o p n  
 o o n o p o o n n no o p o o n  
 o on o n on n p  
 o o n n n o p on n n  
 o on o n p o p o o n  
 n n p p o po n n o p n  
 n o

p o o p o o n o n  
o p n p n po o o o n

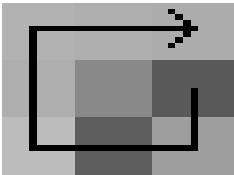


**Fig. 1.** n p o oo n n p n o  
o on o n o n

o o n on n o o p n n o n o n  
on o o n o



**Fig. 2.** n on o  
n o o p on



Feature byte example: 11111010

**Fig. 3.** o n n  
o n n n o p  
o n o

p p o n o on n n  
 o p on o n on  
 n o p n o o on pp o  
 o n n n o oo n n n o n o  
 n o p n n n n o n o n p o  
 o o po n o n o o n  
 n n o n no on n n n  
 o on o o on o o o n  
 o on n n n o p o n o n  
 o on n n n o n on

### 3 Experiments

o o p n po o p o o n  
 n on p o n n o  
 — o o n on p  
 — o p on n n  
 — n o n o

#### 3.1 Assessment

n p n po o p o o n o p on  
 no n n o o n  
 n n on o n on o p o on  
 n p o o n n o p o o p  
 on n n on o o n n n n n  
 p o o p n on n o  
 — n n o n o n o n o n on  
 — n n nn n op on  
 o n n n p n o o o  
 p o n  
 on n p p o pon n  
 o n p o o on p o o p o p  
 on n no o p o o p n on  
 o o no o n  
 n n p o o n p o o n p  
 no o o p on o no on on o  
 n o p on n n o n o  
 p n po n o n n n  
 o n o o n n p

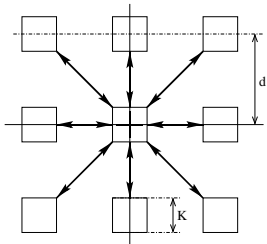


**Fig. 4.**      p   o                      on                      n no

o n      n o o      o p   on                      o o      n  
p o o p                      n n      n      o p   on      o o  
o n      n      on o n o n o      o      o n  
                    n o      n o o      n p      op   on o  
                    o o      n o      n n p      o n n  
o o      o      o p n      o

**3.2 Experimental Results: Octal Coding**

                    o      o      n n n                      o      n o  
o n                      o                      o      p n      n  
*d* p      p                      on o      o n o      oo n n      o n on  
o n *d* on o                      o n o o o p



**Fig. 5.**      o      o o      o n  
                    o o                      n p      *d*

n n      p n      o n *d* on n n  
n on n o o o n      o n p o o n  
                    n on      on      p o n      o

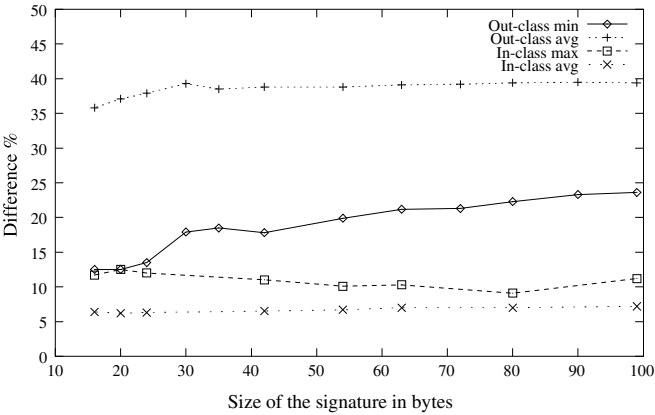
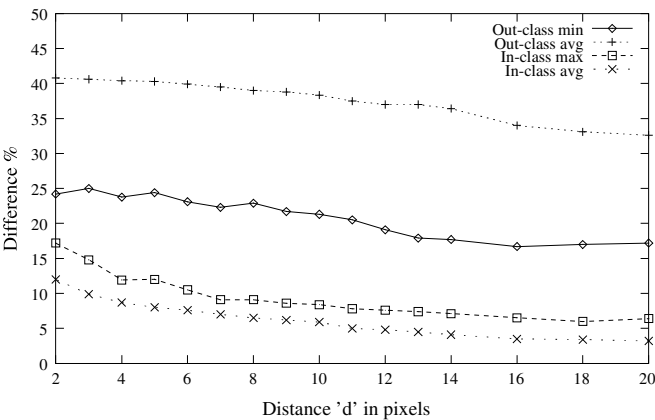
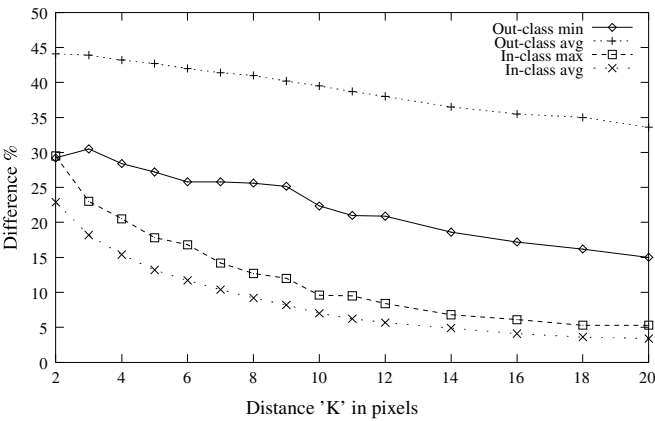


Fig. 6. o n

p o p n n n p o o n n p n  
op o p o o n p n n o  
n op p o n n on p o o n o o  
o p o o n p n o on  
p n n n on p p n n  
p n o p o n n  
n o oo n p n o o  
n p o on n n o n  
o o p o o n n n  
n d on n p o o  
o p n d o n n  
n p o p n p o o  
p o n n n o on n o d n  
o d o o o o oo n o  
o o o d o o oo o on on  
p o n n n o n o o o oo  
p n o p on n d  
on n o o o o o on  
o o n o o n o  
p n p on n n n o o  
n n n n o n  
on n p n o p o n n  
n o o p on p n o  
o n n n n o  
oo n on no o n o n n o o  
n op o o on o n o p p o o



**Fig. 7.** o n nfl n o o  
p n



**Fig. 8.** o n nfl n o o  
p n

4 Conclusions

p o o n pp o o o n p o on p opo  
n n n o n o n n on p op  
pp o p pp op o o n n on p po  
n o n n n o o o n o  
n p o n on o o n n o o

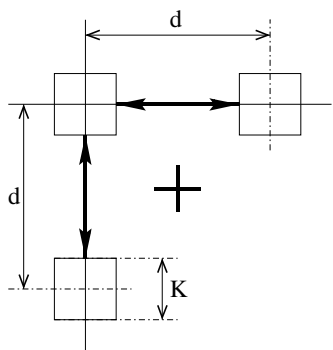


Fig. 9. p o n

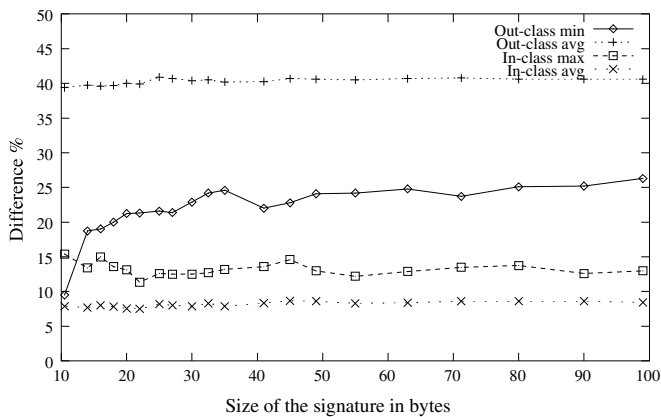


Fig. 10. p o n

o o p o no o p o on n  
n on p o on o o n  
o o o n n o n on n  
o n oo n on n o o n  
o o n p n p o o o n  
n n o n n n on o  
o on o p o n no o  
n o o p on p n o p o o p o  
o n o p o o o n  
o p o n no n p p n o on o  
n o o on o n p o n n on

## References

- [1] I. Rabinovich L. O’Gorman. “secure identification documents via pattern recognition and public-key cryptography”. *IEEE trans. pattern analysis and machine intelligence*, pages 1097–1102, Oct. 1998.

# An Overview of the Isoperimetric Method in Coding Theory (Extended Abstract)

[Invited Paper]

Jean-Pierre Tillich<sup>1</sup> and Gilles Zémor<sup>2</sup>

<sup>1</sup> Université Paris-Sud,

LRI, bâtiment 490, 91405 Orsay, France

<sup>2</sup> École Nationale Supérieure des Télécommunications,

46 rue Barrault, 75634 Paris 13, France

**Abstract.** When decoding a threshold phenomenon is often observed: decoding deteriorates very suddenly around some critical value of the channel parameter. Threshold behaviour has been studied in many situations outside coding theory and a number of tools have been developed. One of those turns out to be particularly relevant to coding, namely the derivation of isoperimetric inequalities for product measures on Hamming spaces. we discuss this approach and derive consequences.

## 1 Background

Let  $C$  be a binary linear code with parameters  $[n, k, d]$ . Denote by  $R = k/n$  its rate and  $\delta = d/n$  its relative minimum distance. We shall be mainly concerned with the asymptotic behaviour of  $C$  so that  $R$  and  $\delta$  should be thought of as fixed quantities as opposed to growing  $n$ . Traditionally, two approaches to coding theory have coexisted over the years. One approach consists of looking for codes with the largest possible minimum distance for a given length and dimension, based on the simple statement that if less than  $d/2$  errors occur then (disregarding complexity issues) the original codeword can always be recovered. The other approach consists of studying the probability of a decoding error after a codeword has been corrupted by some channel. To take one of the most studied examples, the binary symmetric channel with transition probability  $p$ , the probability of a decoding error can be written as :

$$f(p) = 1 - \mu_p(W) = 1 - \sum_{x \in W} p^x (1-p)^{n-x} \quad (1)$$

where  $x$  denotes the Hamming weight of  $x$ . The set of vectors  $W$  is a *decoding region*, i.e. the set of error vectors that will be correctly decoded by a maximum-likelihood decoding scheme. Practitioners and theorists alike have been asking for the behaviour of  $f(p)$  : many results exist on the *typical* behaviour of  $f(p)$  when  $C$  is chosen randomly from an ensemble of codes. In particular, if  $C$  is chosen randomly from all linear codes of (large) length  $n$  and rate  $R$ , then we

know that  $f(p)$  jumps suddenly from almost zero to almost one around the critical value  $\theta = H^{-1}(1 - R)$ , this is essentially Shannon's theorem. The typical value of  $f(p)$  was then further studied in the sixties by Gallager who showed that  $f(p) \sim e^{-nE(R,p)+o(n)}$  for a function  $E(R,p)$  that he computed and that is positive whenever  $p < \theta$ .

One frustrating fact is that the typical critical value  $\theta$  of  $f(p)$  actually equals the typical relative minimum distance  $\delta$ . This means that even if there are many more than  $\delta/2$  corrupted bits, only very few error patterns will actually result in a decoding error. This well-known phenomenon has again been highlighted recently with the progress of iterative decoding techniques, e.g. the advent of turbodecoding. In that case the code  $C$  is chosen among a much more specific and smaller ensemble of codes, and maximum-likelihood decoding is replaced by some suboptimal decoding scheme (this can be seen as replacing the decoding region  $W$  in (1) by a considerably smaller set of vectors). Experiments show that  $f(p)$  still tends to behave in a threshold manner, for impressive values of  $\theta$ , although the codes have poor minimum distance properties. However in these cases, for  $p < \theta$ , the quantity  $f(p)$  does not decay exponentially with the block length  $n$  any more.

Our approach is to try and bridge the gap between the two approaches to coding, the study of the minimum distance and the study of  $f(p)$ . Suppose we are given a code with minimum distance  $d$ : what can we say about  $f(p)$ ? About its threshold behaviour? How can we upperbound  $f(p)$ ? This time we do not ask for typical behaviour but want a result valid for *any* code with minimum distance  $d$ .

## 2 Threshold Effects and the Isoperimetric Method

In 1974 a very elegant result of Margulis [2] went largely unnoticed by the coding community because its implications were not fully apparent.

For any two vectors  $x$  and  $y$  of the Hamming space  $\mathbb{F}_2^n$ , let us write  $x \preceq y$  if for any  $i = 1, 2, \dots, n$   $x_i = 1$  implies  $y_i = 1$ . We shall say that  $W$  is *increasing* if for any  $x \in W$ ,  $x \preceq y$  implies that  $y$  is also in  $W$ .

Margulis introduced the quantity :

$$\begin{aligned} h_W(x) &= 0 && \text{if } x \notin W \\ h_W(x) &= \text{card } \{y \in W, d(x, y) = 1\} && \text{if } x \in W \end{aligned}$$

where  $d(x, y)$  denotes the Hamming distance between  $x$  and  $y$ . Denote by  $\Delta(W)$  the smallest nonzero value of  $h_W(x)$ . Let  $\mu_p$  denote the product probability measure on the Hamming space defined by

$$\mu_p(X) = \sum_{x \in X} p^{\|x\|} (1-p)^{n-\|x\|}$$

for any set of vectors  $X$ . Margulis's theorem is a very general statement about increasing sets.

**Theorem 1 (Margulis 74)** *For any  $\epsilon > 0, \eta > 0$ , there exists  $m > 0$  such that for any increasing set  $W$  satisfying  $\Delta(W) \geq m$ , the set of  $p$ 's for which  $\mu_p(W)$  takes values between  $\epsilon$  and  $1 - \epsilon$  is an interval of length smaller than  $\eta$ .*

This emphasizes the threshold nature of the function  $p \mapsto \mu_p(W)$ . The larger  $\Delta(W)$ , the quicker  $\mu_p(W)$  jumps suddenly from almost zero to almost one.

Why is this relevant to coding ? Because it applies almost immediately to the decoding error probability  $f(p)$  in (1). In this case the decoding region  $W$  is not increasing, but it is a *decreasing* set, i.e.  $x \in W$  and  $y \notin x$  imply  $y \notin W$  : Margulis's theorem will also apply. Furthermore, the quantity  $\Delta(W)$  is directly dependent on the minimal distance of the code  $d$ , we have namely  $\Delta(W) \geq d/2$ . The consequence is that the decoding error probability  $f(p)$  behaves in a threshold manner, i.e. jumps suddenly from almost zero to almost one, and that the "jump" narrows as the minimum distance grows.

Deriving such a result is not straightforward and Margulis's method is especially interesting. It relies on the following identity, later to become known in percolation theory as Russo's identity, which states that for any increasing set  $W$  :

$$\frac{d\mu_p(W)}{dp} = \frac{1}{p} \int_W h_W(x) d\mu_p(x). \quad (2)$$

Margulis then goes on to lower bound the quantity  $\int_W h_W(x) d\mu_p(x)$  by a function of  $\mu_p(W)$ . Integrating the resulting differential inequality then yields the threshold behaviour. The method can be named isoperimetric because the integral  $\int_W h_W(x) d\mu_p(x)$  can be thought of as a measure of the "boundary" of  $W$  and is lower bounded by a function of its "volume"  $\mu_p(W)$ .

Margulis's theorem was made much more explicit by Talagrand [3] who showed that the estimation of  $\mu_p(W)$  can be made more precise by considering a modified measure of the boundary of  $W$ , namely  $\int \overline{h_W} d\mu_p$ .

Talagrand's isoperimetric inequalities were refined by Bobkov and Goetze [1], and improved again in [4]. After integration, these inequalities yield the following result for increasing sets.

**Theorem 2 (Tillich Zémor 99)** *Let  $W$  be an increasing set of vectors of  $\mathbb{F}_2^n$ , and let  $\Delta = \Delta(W)$ . Let  $\theta$  be defined by  $\mu_\theta(W) = 1/2$ . Then  $\mu_p(W)$  satisfies :*

$$\mu_p(W) \leq \Phi \left( \frac{\overline{\Delta}}{2\Delta} \left( \frac{1}{1 - \ln \theta} - \sqrt{-\ln p} \right) \right) \quad \text{for } 0 < p < \theta \quad (3)$$

$$\mu_p(W) \leq \Phi \left( \frac{\overline{\Delta}}{2\Delta} \left( \frac{1}{1 - \ln \theta} - \sqrt{-\ln p} \right) \right) \quad \text{for } \theta < p < 1. \quad (4)$$

where  $\Phi(x) = \frac{1}{2\pi} \int_{-\infty}^x e^{-t^2/2} dt$ .

Applied to coding, this yields the following theorem.

**Theorem 3 (Tillich-Zémor 99)** *Let  $C$  be a binary linear code with minimum distance  $d$  and any length. Over the binary symmetric channel with transition probability  $p$ , the probability of decoding error  $f(p)$  associated to  $C$  satisfies :*

$$\begin{aligned} f(p) &= 1 - \Phi \left[ \frac{d}{2} \left( \sqrt{-\ln(1-\theta)} - \sqrt{-\ln(1-p)} \right) \right] & \text{for } 0 < p < \theta \\ f(p) &= 1 - \Phi \left[ \frac{d}{2} \left( \sqrt{-\ln(1-\theta)} - \sqrt{-\ln(1-p)} \right) \right] & \text{for } \theta < p < 1. \end{aligned}$$

where  $\theta$  is defined by  $f(\theta) = 1/2$ .

This theorem makes the threshold behaviour of  $f(p)$  very precise. Note that for fixed  $a < 0$  and growing  $d$  the quantity  $\Phi(a/\frac{d}{2})$  is equivalent to  $\frac{1}{a} \frac{1}{2\pi d} e^{-da^2/2}$ , so that theorem 3 really gives an upper bound of the form

$$f(p) \leq \exp(-dg(\theta, p))$$

where  $g(\theta, p) > 0$  for  $p < \theta$  : in other words  $f(p)$  is exponentially small in  $d$ . In particular, families of codes with minimal distance growing linearly with their length  $n$  have a probability of decoding error which decreases exponentially with  $n$ , as long as  $\theta - p$  stays bounded below by some  $\varepsilon > 0$ . We now know that this holds for *all* such codes, not just that it is typical behaviour.

### 3 Locating the Threshold

The isoperimetric method gives precise results on the behaviour of the decoding error probability  $f(p)$  given  $d$  but does not say anything about the whereabouts of the threshold probability  $\theta$ . As mentioned earlier, randomly chosen codes with large length have  $\theta = \delta = d/n$ , but what is the (asymptotic) situation for any code with prescribed relative distance  $\delta$  ? More precisely, denoting by  $\theta(C)$  the threshold value for a code  $C$ , we would like to determine

$$\Theta = \liminf \theta(C)$$

where the  $\liminf$  is defined over any sequence enumerating the set of all codes  $C$  such that  $d \geq \delta n$ . What is the best lower bound on  $\Theta$  as a function of  $\delta$  ? This is an interesting open question. It should be clear that we must have  $\Theta \geq \delta/2$ . If it were true that  $\Theta = \delta$ , this would imply that the Varshamov-Gilbert bound is tight. The best lower bound on  $\Theta$  known to us makes use of averaging arguments together with bounds on the highest possible rate of constant weight codes [4] : here are some numerical values.

**Table 1.**  $\Theta$  as a function of  $\delta$

$\delta$	0.1	0.2	0.3	0.35	0.4	0.45	0.5
lower bound on $\Theta$	0.053	0.123	0.212	0.267	0.330	0.385	0.5

## 4 The Erasure Channel

Theorem 2 is very general and should find applications in a variety of situations. Its applications to coding are especially interesting in the context of the erasure channel. Let  $C$  be again a binary linear code with parameters  $[n, k, d]$ . This time, when a codeword of  $C$  is transmitted its symbols are erased independently with probability  $p$ . Let  $e \in \mathbb{F}_2^n$  be the *erasure vector*, i.e. the characteristic vector of the set of erased positions. The probability  $f(p)$  that we are now interested in is the probability that the initial codeword cannot be recovered from the set of received symbols : it is straightforward to check that this happens exactly when  $c \cdot e \neq 0$  for some nonzero codeword  $c$ . We have therefore :

$$f(p) = \mu_p(W)$$

where  $W$  now stands for the set of vectors  $x$  for which there exists  $c \in C$ ,  $c \cdot x \neq 0$  such that  $c \cdot e = 0$ . Every vector in  $W$  has weight at least  $d$ , from which we have the well-known fact that  $C$  can always correct up to  $d - 1$  erasures. But it might very well be true (actually it is true, this is our point) that  $C$  can, with high probability, correct many more erasures.

It is clear that  $W$  is an increasing set of vectors. Furthermore, it is not difficult to show that, because  $C$  is linear,  $\Delta(W) = d$ . Therefore the isoperimetric method applies, and theorem 2 translates directly into a theorem of a nature similar to that of theorem 3 (see [4]).

Actually, Margulis's initial motivation for deriving his theorem was the study of this function  $f(p)$  in the particular situation when  $C$  is the cocycle code of a graph. In this case  $f(p)$  represents the probability that a random set of edges disconnects the graph.

As in the case of the binary symmetric channel, we would like to lower bound the threshold  $\theta$  (defined by  $f(\theta) = 1/2$ ) by a function of  $\delta$ . Defining again  $\Theta = \liminf \theta(C)$  where  $C$  runs over all codes such that  $d \geq \delta n$ , we have trivially that  $\Theta \geq \delta$ . But interestingly, in this case the isoperimetric approach can be pushed further to yield nontrivial lower bounds on  $\Theta$ .

The idea is to define the sequence of sets

$$W_1 = W \supset W_2 \supset \dots \supset W_t \supset \dots$$

where  $W_t$  is the set of vectors  $x$  such that

$$C_x = \{c \in C : c \cdot x \neq 0\}$$

is a subcode of  $C$  of dimension  $t$ . The threshold probabilities associated to each  $W_t$  form a sequence

$$\theta_1 = \theta \quad \theta_2 \quad \dots \quad \theta_t \quad \dots$$

The isoperimetric method will show that the differences  $\theta_{t+1} - \theta_t$  must tend to zero as the minimum distance tends to infinity. We can then argue that a code of length  $\theta_t n$ , dimension  $t$ , and minimum distance  $d$  must exist, so that these parameters must not contradict existing bounds. This argument gives [5,4]

$$\Theta \quad 2\delta$$

and can be pushed further to yield improved lower bounds : this is the object of forthcoming work.

## References

1. Bobkov, S., Goetze, F. (1996) Discrete Isoperimetric and Poincaré-type inequalities. Technical report SFB 343 University of Bielefeld 96-086. <ftp://ftp.mathematik.uni-bielefeld.de/pub/papers/sfb343/pr96086.ps.gz>
2. Margulis, G. (1974) Probabilistic characteristics of graphs with large connectivity. *Problemy Peredachi Informatsii*. **10**, 101–108
3. Talagrand, M. (1993) Isoperimetry, logarithmic Sobolev inequalities on the discrete cube, and Margulis' graph connectivity theorem. *Geometric and Functional Analysis*. **3**, 295–314.
4. Tillich, J-P., Zémor, G. (1999) Discrete inequalities and the probability of a decoding error. Submitted to *Combinatorics, Probability & Computing*. <http://www.infres.enst.fr/zemor/isoperimetric.ps>
5. Zémor, G. (1994) Threshold effects in codes. In *Algebraic coding*, Springer-Verlag, LNCS 781 278–286.

# Rectangular Basis of a Linear Code

Johannes Maucher<sup>1</sup>, Vladimir Sidorenko<sup>2</sup>, and Martin Bossert<sup>1</sup>

<sup>1</sup> Department of Information Technology, University of Ulm,  
Albert-Einstein-Allee 43, 89081 Ulm, Germany,  
`joma,boss@it.e-technik.uni-ulm.de`

<sup>2</sup> Institute for Information Transmission Problems,  
Russian Academy of Science,  
B.Karetnyi per.19 101447, Moscow GSP-4, Russia,  
`sid@iitp.ru`

**Abstract.** A rectangular code is a code for which there exists an unique minimal trellis. Such a code can be considered to be an algebraically closed set under the rectangular complement operation. The notions of rectangular closure and basis were already defined. In this paper we represent a method to construct a rectangular basis of a linear code from a given linear basis.

## 1 Introduction

Each code  $C$  can be represented in a trellis  $T(C)$ . This trellis representation is applied in decoding algorithms, for example in the Viterbi decoding algorithm. For a given code there exists a large variety of corresponding trellises. Obviously, for most applications a trellis with a minimal complexity is preferred, however there exists different complexity measures. It was shown in [3] that there exists a class of codes for which there exists a unique minimal trellis, i.e. a trellis which minimizes all ordinary complexity measures. This set of codes is called the set of rectangular codes. It can easily be shown that each linear code is a rectangular code. Hence, in most of the previous works people investigated which nonlinear codes are rectangular. In recent works [8], [7] the algebraic structure of rectangular codes is studied. It is shown in these papers that for any nonrectangular code there exists a unique rectangular closure. Moreover, an algorithm is proposed which computes for each rectangular code a rectangular basis. For a given nonrectangular code the trellis of the corresponding rectangular closure has a smaller complexity, than the trellis of the nonrectangular code itself. Therefore the decoding complexity decreases if a nonrectangular code is decoded in the trellis of its rectangular closure. Moreover, as shown in [9], for some iterative decoding algorithms which use a set of low weight codewords of a linear code, complexity decreases and performance increases if the rectangular closure of these low weight codewords is used. The merit of a rectangular basis is that it provides a quite compact description of a rectangular code, i.e. in applications in which a rectangular set must be stored on a device it is sufficient to store only its rectangular basis and generate the whole set from this basis, whenever it is needed.

In this paper we investigate the relation between rectangular bases and linear bases of linear codes. In particular we show how a rectangular basis of a linear code can be derived from its generator matrix in trellis oriented form.

## 2 Notations and Definitions

### 2.1 Trellis Representation of a Code $C$

A trellis  $T = (V, E, A)$  is an edge labeled directed graph. In a trellis of length  $n$  the set of vertices is  $V = V_0 \text{ --- } V_n$  and the set of edges is  $E = E_1 \text{ --- } E_n$ . Each edge  $e \in E_i$  connects a vertex  $v \in V_{i-1}$  with a vertex  $v' \in V_i$ . The initial vertex of  $e$  is then  $i(e) = v$  and its final vertex is  $f(e) = v'$ . The edge  $e$  is said to be incident from  $i(e) = v$  and incident to  $f(e) = v'$ . A path of length  $l$  consists of a sequence of edges  $[e_{i_1} \dots e_{i_l}]$ , such that  $f(e_{i_j}) = i(e_{i_{j+1}})$  for all  $j = 1, \dots, l-1$ . Each edge  $e \in E_i$  is labeled by an element from the finite set  $A_i$ . Then each path of length  $n$  from a vertex in  $V_0$  to a vertex in  $V_n$  is labeled by an element from the cartesian product  $A = A_1 \text{ --- } A_n$ . In a trellis  $T(C)$  of a code  $C$  there exists only one vertex  $v_r$  in  $V_0$  and only one vertex  $v_g$  in  $V_n$ . For each codeword  $\mathbf{c} \in C$  there exists a path  $pa(\mathbf{c})$  from  $v_r$  to  $v_g$  which is labeled by  $\mathbf{c}$ . Usually trellises  $T(C)$  are considered in which there exists for each  $\mathbf{c}$  only one path  $pa(\mathbf{c})$  labeled by  $\mathbf{c}$ . Among all possible trellises  $T(C)$  of a given code  $C$ , the minimal trellis of  $C$  is the one which minimizes the number of vertices in each  $V_i$  simultaneously [5]. There exists an unique minimal trellis for  $C$ , iff  $C$  is rectangular [3].

### 2.2 Rectangular Codes

In general a rectangular code of length  $n$  can be considered to be a subset of the cartesian product  $A = A_1 \text{ --- } A_n$  of arbitrary finite sets  $A_i$ . However, since we consider in this paper linear codes we restrict throughout this paper rectangular codes to be subsets of the  $n$ -dimensional vector space  $GF(q)^n$ . Any codeword  $\mathbf{c} = (c_1, \dots, c_n) \in C$  can be partitioned in an arbitrary depth  $t = 1, \dots, n$  into its  $t$ -head  $p = (c_1, \dots, c_t)$  and its  $t$ -tail  $f = (c_{t+1}, \dots, c_n)$ . We denote by  $\text{--}_t(C)$  the set of all  $t$ -heads and by  $\mathcal{F}_t(C)$  the set of all  $t$ -tails of code  $C$ .

**Definition 1** Let  $p_1, p_2 \in \text{--}_t(C)$  and  $f_1, f_2 \in \mathcal{F}_t(C)$ . Then  $C$  is called  $t$ -rectangular, iff

$$(p_1, f_1), (p_1, f_2), (p_2, f_1) \in C \text{ implies } (p_2, f_2) \in C. \quad (1)$$

If  $C$  is  $t$ -rectangular in all depths  $t = 1, \dots, n$ , then  $C$  is rectangular.

**Example 1** The code  $C = \{110, 101, 111\}$  is 1-rectangular, but not 2-rectangular, because the vector  $v = 100$  is not contained in  $C$ .

All linear codes are rectangular because they are closed under addition and subtraction of codewords: For the codewords  $\mathbf{c}_1 = (p_1, f_1)$ ,  $\mathbf{c}_2 = (p_1, f_2)$  and  $\mathbf{c}_3 = (p_2, f_1)$  of the linear code  $C$ , the vector

$$\mathbf{c}_4 = \mathbf{c}_2 - \mathbf{c}_1 + \mathbf{c}_3 = (p_1, f_2) - (p_1, f_1) + (p_2, f_1) = (p_2, f_2)$$

is again a codeword of  $C$ . Thus the defining relation (1) is fulfilled for all depths  $t = 1, \dots, n$ .

For any set of vectors  $Y \subseteq GF(q)^n$  the *rectangular closure*  $S = R(Y)$  is defined to be the smallest rectangular set which contains  $Y$ . The rectangular closure of a given set  $Y$  is unique.  $R(Y)$  is closed under the rectangular complement operation, which is defined as follows:

**Definition 2** Let  $p_i$  be the  $t$ -head and  $f_i$  be the  $t$ -tail of vector  $\mathbf{c}_i$ . Let  $-\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3-$  be a set of three vectors with  $f_1 = f_2$  and  $p_2 = p_3$ . Then the  $t$ -rectangular complement

$$\mathbf{z} := r_t(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$$

of  $-\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3-$  is  $\mathbf{z} = (p_1, f_3)$ .

Note that the rectangular complement of a set of three vectors is defined, iff within this set there exists a pair of vectors which has the same  $t$ -head and a pair of vectors which has the same  $t$ -tail. If for a given set of three vectors  $-\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3-$  the rectangular complement is defined in depth  $t$  and depth  $l$ , then  $r_t(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3) = r_l(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ . Therefore the depth index in the rectangular complement operation can be omitted. A rectangular closed set  $S$ , together with the rectangular complement operation  $r$  constitutes an algebra  $\mathcal{A} = \langle S; r \rangle$ . Strictly speaking it is a partial algebra [2] since the operation  $r$  is not defined on all triples of vectors.

**Example 2** The rectangular complement of the three vectors of  $C$  from Example 1 is not defined in depth  $t = 1$ . However in depth  $t = 2$  the rectangular complement is defined:

$$r(110, 101, 111) = 100.$$

The new set  $S = C - 100-$  is rectangular, i.e.  $S = R(C)$  is the rectangular closure of  $C$ .

A set of vectors  $Y$  is said to be a *rectangular independent* set if none of the vectors  $\mathbf{y}_i \in Y$  is contained in the rectangular closure of the other vectors in  $Y$ :

$$\mathbf{y}_i \notin R(Y - \mathbf{y}_i).$$

If  $S$  is the rectangular closure of  $Y$ , then  $Y$  is said to be a *generating set* of  $S$ . A generating set of  $S$ , which is rectangular independent is a *basis* of  $S$ . In the sequel we denote a bases by  $B$  and its elements by  $\mathbf{b}$ .

### 3 Construction of a Rectangular Basis

Throughout this section  $C$  denotes a rectangular code in general, i.e.  $C$  is not restricted to be linear. We represent an algorithm which constructs a rectangular basis  $B(T)$  of code  $C$  in the minimal trellis  $T(C)$ . This algorithm was introduced in [8]. The proof of Theorem 1 can be found in [4].

#### Coloring Algorithm:

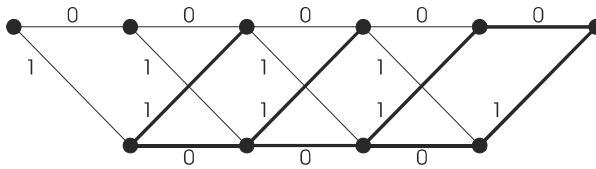
1. For each vertex  $v$  in the minimal trellis  $T(C)$  all edges, incident to  $v$ , except one edge, must be colored. All edges, incident to the goal vertex  $v_g$  must be colored. Set  $B(T) = \emptyset$ .
2. For each colored edge  $e$  construct a path in  $T(C)$  that goes from the root vertex  $v_r$  to the goal vertex  $v_g$  through the edge  $e$  and comes to the vertex  $v = i(e)$  through noncolored edges. Join the codeword corresponding to this path to the set  $B(T)$ .

**Theorem 1** If  $T = (V, E, A)$  is a trellis of a rectangular code  $C$ , then  $G(T)$  is a generating set of  $C$  and

$$|B(T)| = |E| - |V| + 2.$$

If in addition the trellis  $T$  is the minimal, then  $B(T)$  is a basis of  $C$ .

**Example 3** For the parity check code  $C(5, 4, 2)$ , the minimal trellis and a possible coloring according to item 1 of the coloring algorithm is shown in the picture below, where ‘colored’ edges are printed bold.



From this colored trellis one can determine, according to item 2 of the coloring algorithm, for example the basis  $B = \{-11000, 10100, 01100, 01010, 00110, 00101, 00011, 00000\}$ .

### 4 Construction of a Rectangular Basis for Linear Codes

The drawback of the coloring algorithm is that it constructs a rectangular basis in the minimal trellis and the construction of such a trellis may be quite complex for large codes. In the sequel we will represent a method to construct a rectangular basis of a linear code directly from its generator matrix. The merit of this new method is, that it is less complex than the coloring algorithm and it provides a

better understanding of the relation between rectangular and linear bases. In [1] Forney introduced the trellis oriented generator matrix. From the trellis oriented generator matrix of a code  $C$  one can directly determine some properties of the minimal trellis  $T(C)$ . In particular there exists an efficient method to construct the minimal trellis from the rows of this matrix [3]. On the other side we know a method to get a rectangular basis in the minimal trellis by the coloring algorithm. We will show how these two methods can be combined to construct a rectangular basis from the trellis oriented generator matrix such that the intermediate step of constructing a minimal trellis is not necessary.

#### 4.1 Generator Matrix in Trellis Oriented Form

For a given codeword  $\mathbf{c} = (c_1, c_2, \dots, c_n) \in C$  the *left index*  $\mathcal{L}(\mathbf{c})$  is defined to be the smallest depth  $i$  for which  $c_i = 0$  and the *right index*  $-\mathbf{c}$  is defined to be the highest depth  $i$  for which  $c_i = 0$ . For example the left index of  $\mathbf{c} = (0100100)$  is  $\mathcal{L}(\mathbf{c}) = 2$  and its right index is  $-\mathbf{c} = 5$ . Vector  $\mathbf{c}$  is said to be *active* within the interval  $[\mathcal{L}(\mathbf{c}), \dots, -\mathbf{c} - 1]$  and *passive* in all depths outside this interval.

**Definition 3** Let  $G$  be the generator matrix of a linear code  $C$ , and denote the  $i$ .th row of  $G$  by  $\mathbf{g}_i$ . Then  $G$  is said to be in *trellis oriented form*, if all pairs of rows  $\mathbf{g}_i, \mathbf{g}_j$  have distinct left and distinct right indices:

$$\mathcal{L}(\mathbf{g}_i) \neq \mathcal{L}(\mathbf{g}_j) \quad , \quad -(\mathbf{g}_i) \neq -(\mathbf{g}_j) \quad (2)$$

**Example 4** A generator matrix of a parity check code  $C(5, 4, 2)$  is

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Since (2) is fulfilled,  $G$  is in trellis oriented form.

In [1] it is mentioned, that for each linear code there exists a trellis oriented generator matrix.

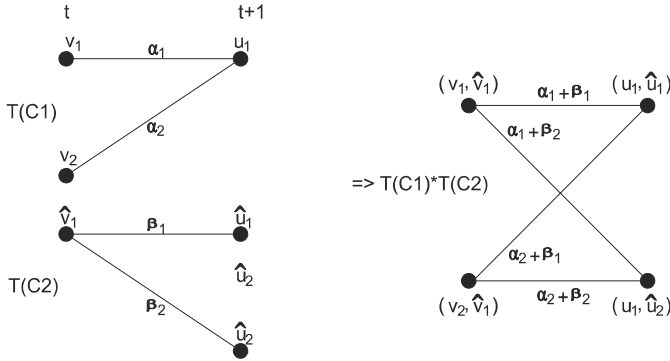
#### 4.2 Minimal Trellis Construction Based on the Shannon Product

Given a pair of linear codes  $C_1, C_2$  of the same length and corresponding code trellises  $T(C_1), T(C_2)$  the trellis  $T(C)$  of their sum

$$C = C_1 + C_2 = -\mathbf{c}_1 + \mathbf{c}_2 \quad - \quad \mathbf{c}_1 - C_1, \mathbf{c}_2 - C_2 -$$

can be obtained by computing the Shannon product  $T(C_1) - T(C_2) = T(C)$ , which is defined as follows:

**Definition 4** Let  $V_t$  and  $\widehat{V}_t$  be the set of vertices in depth  $t$  of trellises  $T(C_1) = (V, E, A)$  and  $T(C_2) = (\widehat{V}, \widehat{E}, A)$ , respectively. The vertices of the Shannon product  $T(C) = T(C_1) \cdot T(C_2)$  are then marked by pairs  $(v, \widehat{v})$  with  $v \in V_t$  and  $\widehat{v} \in \widehat{V}_t$ . In  $T(C)$  a vertex  $(v, \widehat{v})$  in depth  $t$  is connected to a vertex  $(u, \widehat{u})$  in depth  $t+1$ , iff in  $T(C_1)$  vertex  $v \in V_t$  is connected to vertex  $u \in V_{t+1}$  and in  $T(C_2)$  vertex  $\widehat{v} \in \widehat{V}_t$  is connected to vertex  $\widehat{u} \in \widehat{V}_{t+1}$ . If  $\alpha$  is the label of the edge which connects  $v$  and  $u$  in  $T(C_1)$ , and  $\beta$  is the label of the edge which connects  $\widehat{v}$  and  $\widehat{u}$  in  $T(C_2)$ , then the label of the edge which connects vertex  $(v, \widehat{v})$  with vertex  $(u, \widehat{u})$  in  $T(C)$  is  $\alpha + \beta$  :



Let  $\mathbf{g}_i$  be a row of a generator matrix of a linear  $(n, k)$ -code  $C$ , defined over  $GF(q)$ . We denote by  $T(\mathbf{g}_i)$  the minimal trellis of a subcode

$$C_i = \langle \mathbf{g}_i \rangle = \langle s\mathbf{g}_i \mid s \in GF(q) \rangle,$$

spanned by  $\mathbf{g}_i$ . The minimal trellis  $T(C)$  can be constructed from its trellis oriented generator matrix  $G$  by a stepwise calculation of the Shannon product of subcode trellises:

**Theorem 2** Let  $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$  be the rows of the generator matrix  $G$  of a linear code  $C$ . Then the Shannon product  $T(\mathbf{g}_1) \cdot \dots \cdot T(\mathbf{g}_k)$  is a minimal trellis of  $C$ , iff  $G$  is in trellis oriented form.

This Theorem is proved in [3] and [6].

## 5 Construction of a Rectangular Basis from the Trellis Oriented Generator Matrix

### 5.1 Subcodes of Dimension $k = 1$

The minimal trellis  $T(\mathbf{g}_i)$  has the property that in all depths  $l$  within the two intervals  $[1, \dots, \mathcal{L}(\mathbf{g}_i) - 1]$  and  $[-(\mathbf{g}_i), \dots, n]$  all components of vector  $\mathbf{g}_i$  are zero. Therefore in these depths  $l$  also all components  $c_l$  of all codewords in  $C_i$  are zero. From this follows property  $P1$  and from  $P1$  follow properties  $P2, P3$  and  $P4$  of the minimal trellis  $T(\mathbf{g}_i)$ .

- P1** All codewords of  $C_i$  have the same  $(\mathcal{L}(\mathbf{g}_i) - 1)$ -head, but distinct  $l$ -heads for  $l - \mathcal{L}(\mathbf{g}_i)$ . All codewords of  $C_i$  have the same  $-(\mathbf{g}_i)$ -tail but distinct  $l$ -tails for  $l < -(\mathbf{g}_i)$ .
- P2** All paths  $pa(\mathbf{c})$ ,  $\mathbf{c} \in C_i$  go through the same vertex in depth  $l$ , if  $\mathbf{g}_i$  is passive in depth  $l$ .
- P3** For all pairs  $\mathbf{c}_a, \mathbf{c}_b$  of distinct vectors from  $C_i$  the paths  $pa(\mathbf{c}_a)$  and  $pa(\mathbf{c}_b)$  go through distinct vertices in depth  $l$ , if  $\mathbf{g}_i$  is active in depth  $l$ .
- P4** In depth  $l = -(\mathbf{g}_i)$  all  $q$  distinct paths  $pa(\mathbf{c})$  of codewords from  $C_i$  merge into a single vertex, denoted by  $vr(i)$ .

Note that  $vr(i)$  is the only vertex in  $T(\mathbf{g}_i)$  in which merges more than one path - in particular  $q$  paths. The coloring in this trellis can be chosen for example such that all edges incident to  $vr(i)$  which are labeled by a nonzero element from  $GF(q)$  are colored. The set of  $q - 1$  basis codewords  $\mathbf{b}$  which belong to these colored edges in depth  $l = -(\mathbf{g}_i)$  is then the set of all nonzero codewords in  $C_i$ . However, this set is not the complete basis, since one must also assign a basis codeword  $\mathbf{b}$  to the colored edge  $e$  in depth  $l = n$ , which is incident to  $v_g$ . Since this codeword must correspond to a noncolored path from  $v_r$  to  $v = i(e)$ , it can only be the allzero codeword from  $C_i$ . This proves the following Theorem:

**Theorem 3** *The rectangular basis of a linear code  $C$  of dimension  $k = 1$  is  $B = C$ .*

## 5.2 Linear Code of Dimension $k$

Let us now investigate how to determine colored edges, i.e. basis vectors which are assigned to colored edges, directly from the trellis oriented generator matrix of a linear code. By Theorem 2 the minimal trellis  $T(C)$  of such a code is the Shannon product of the minimal trellises of the subcodes:

$$T(C) = T(\mathbf{g}_1) \text{ --- } T(\mathbf{g}_k).$$

From Definition 4 follows that in trellis  $T(C) = T(C_i) \text{ --- } T(C_j)$  there exists a vertex in which merges more than one edge in depth  $t$ , iff in at least one of the trellises  $T(C_1), T(C_2)$  there exists a vertex in depth  $t$ , in which merges more than one edge. This means that in  $T(C) = T(\mathbf{g}_1) \text{ --- } T(\mathbf{g}_k)$  there exist vertices in which merge more than one edge only in depths  $-(\mathbf{g}_1), -(\mathbf{g}_2), \dots, -(\mathbf{g}_k)$ . W.l.o.g. we assume that the rows of the trellis oriented generator matrix are ordered such that  $\mathcal{L}(\mathbf{g}_i) < \mathcal{L}(\mathbf{g}_{i+1})$  and  $-(\mathbf{g}_i) < -(\mathbf{g}_{i+1})$  for all  $i = 1, \dots, k - 1$ . In the sequel we will determine the codewords, which correspond to edges, which merge together in one of the depths  $-(\mathbf{g}_i)$ .

We define  $\overline{G}_i$  to be the matrix, which consists of all rows of  $G$ , except row  $\mathbf{g}_i$ . The code generated by  $\overline{G}_i$  is then the complement of subcode  $C_i$  in  $C$ :

$$\overline{C}_i = \langle \mathbf{g}_1, \dots, \mathbf{g}_{i-1}, \mathbf{g}_{i+1}, \dots, \mathbf{g}_k \rangle$$

For a fixed codeword  $\bar{\mathbf{c}} - \bar{C}_i$  we define a coset  $C_i(\bar{\mathbf{c}})$  of  $C_i$  as follows:

$$C_i(\bar{\mathbf{c}}) = -\mathbf{c} + \bar{\mathbf{c}} \quad - \quad \mathbf{c} - C_i = -s\mathbf{g}_i + \bar{\mathbf{c}} \quad - \quad s - GF(q). \quad (3)$$

Using these definitions we can generalize properties  $P1$  to  $P4$  as follows:

- Q1** All codewords of  $C_i(\bar{\mathbf{c}})$  have the same  $(\mathcal{L}(\mathbf{g}_i) - 1)$ -head but distinct  $l$ -heads for  $l - \mathcal{L}(\mathbf{g}_i)$ . All codewords of  $C_i(\bar{\mathbf{c}})$  have the same  $-(\mathbf{g}_i)$ -tail but distinct  $l$ -tails, for  $l < -(\mathbf{g}_i)$ .
- Q2** All paths  $pa(\mathbf{c}), \mathbf{c} - C_i(\bar{\mathbf{c}})$  go through the same vertex in depth  $l$ , if  $\mathbf{g}_i$  is passive in depth  $l$ .
- Q3** For all pairs  $\mathbf{c}_a, \mathbf{c}_b$  of distinct vectors from  $C_i(\bar{\mathbf{c}})$  the paths  $pa(\mathbf{c}_a)$  and  $pa(\mathbf{c}_b)$  go through distinct vertices in depth  $l$ , if  $\mathbf{g}_i$  is active in depth  $l$ .
- Q4** In depth  $l = -(\mathbf{g}_i)$  all  $q$  distinct paths  $pa(\mathbf{c})$  of codewords from  $C_i(\bar{\mathbf{c}})$  merge into a single vertex, denoted by  $vr(i, \bar{\mathbf{c}})$ .

We will now determine the set of distinct vertices  $vr(i, \bar{\mathbf{c}})$  in depth  $l_i = -(\mathbf{g}_i)$ , in particular the corresponding codewords  $\bar{\mathbf{c}} - \bar{C}_i$  whose paths  $pa(\bar{\mathbf{c}})$  go through vertex  $vr(i, \bar{\mathbf{c}})$ .

In each depth  $l_i = -(\mathbf{g}_i)$  the generator matrix  $G$  can be partitioned in submatrices  $G^{(l_i)}$  and  $\bar{G}^{(l_i)}$  as follows:  $G^{(l_i)}$  is defined to be the matrix, which consists of the  $k^{(l_i)}$  rows of the generator matrix  $G$ , which are passive in depth  $l_i$  and  $\bar{G}^{(l_i)}$  is defined to be the matrix, which consists of the  $\bar{k}^{(l_i)} = k - k^{(l_i)}$  rows of the generator matrix  $G$ , which are active in depth  $l_i$ . The codes generated by  $G^{(l_i)}$  and  $\bar{G}^{(l_i)}$  are denoted by  $C^{(l_i)}$  and  $\bar{C}^{(l_i)}$ , respectively. In the special case  $\bar{k}^{(l_i)} = 0$  we define  $\bar{C}^{(l_i)}$  to contain only the all-zero codeword.

For a fixed codeword  $\mathbf{u} - \bar{C}^{(l_i)}$  we define a coset  $C^{(l_i)}(\mathbf{u})$  of  $C^{(l_i)}$  as follows:

$$C^{(l_i)}(\mathbf{u}) = -\mathbf{a} + \mathbf{u} \quad - \quad \mathbf{a} - C^{(l_i)}. \quad (4)$$

Note that in (4) all vectors  $\mathbf{a} - C^{(l_i)}$  are generated by all rows of  $G$ , which are passive in depth  $l_i$ . Therefore we have property  $S1$ . The vector  $\mathbf{u}$  in (4) is a vector generated by active rows of  $G$  in depth  $l_i$ , which yields  $S2$ . Combining  $S1$  and  $S2$  yields  $S3$ .

- S1** In depth  $l_i = -(\mathbf{g}_i)$  all paths  $pa(\mathbf{c}), \mathbf{c} - C^{(l_i)}(\mathbf{u})$  go through the same vertex.
- S2** For all pairs  $\mathbf{u}_a, \mathbf{u}_b$  of distinct vectors from  $\bar{C}^{(l_i)}$ , the paths  $pa(\mathbf{c}), \mathbf{c} - C^{(l_i)}(\mathbf{u}_a)$  go through distinct vertices than the paths  $pa(\mathbf{c}), \mathbf{c} - C^{(l_i)}(\mathbf{u}_b)$ .
- S3** In depth  $l$  there exist  $q^{\bar{k}^{(l_i)}}$  distinct vertices. Each path  $pa(\mathbf{u}), \mathbf{u} - \bar{C}^{(l_i)}$  goes through a distinct vertex in depth  $l_i$ .

From property  $S3$  we know that in each depth  $l_i = -(\mathbf{g}_i)$ ,  $i = -1, \dots, k-$  there exist  $q^{\bar{k}^{(l_i)}}$  distinct vertices  $vr(i, \mathbf{u})$ , and from  $Q4$  we know that in each of these vertices merge  $q$  distinct edges. These  $q$  distinct edges belong to the  $q$  distinct paths  $pa(\mathbf{c}), \mathbf{c} - C_i(\mathbf{u})$ . W.l.o.g. the coloring of the edges which merge

into vertex  $vr(i, \mathbf{u})$  can be chosen such that all  $q-1$  edges which belong to paths  $pa(\mathbf{c}), \mathbf{c} - C_i(\mathbf{u}) - \mathbf{u}$  are colored. This means that all vectors  $\mathbf{c} - C_i(\mathbf{u}) - \mathbf{u}$  must belong to the basis. If we apply the defined coloring to all  $q^{k(l_i)}$  distinct vertices  $vr(i, \mathbf{u})$  in depth  $l_i = -(\mathbf{g}_i)$  the set of basis vectors  $B_i$ , i.e. the basis vectors which belong to colored edges in depth  $i$  is:

$$\begin{aligned} B_i &= -C_i(\mathbf{u}) - \mathbf{u} - \mathbf{u} - \overline{C}^{(l)} - \\ &= -s\mathbf{g}_i + \mathbf{u} - s - GF(q) - 0 - \text{and } \mathbf{u} - \overline{C}^{(l)} - \end{aligned}$$

where  $\overline{C}^{(l)}$  is the code generated by all rows of  $G$ , which are active in depth  $l_i = -(\mathbf{g}_i)$ . Taking into account that in depth  $l = n$  all edges, incident to the goal vertex  $v_g$  must be colored and the basisvector, which corresponds to this additional colored edge can be chosen to be the all zero codeword  $\mathbf{0}$ , the rectangular basis  $B$  of code  $C$  is

$$B = -\mathbf{0} - \bigcup_{i=1, \dots, k} B_i.$$

Thus the basis  $B$  can be determined directly from the rows of the trellis oriented generator matrix, without constructing the minimal trellis  $T(C)$ .

**Example 5** From the generator matrix of the parity check code  $C(5, 4, 2)$ , represented in Example 4 we obtain the sets  $B_1 = -11000, 10100-$ ,  $B_2 = -01100, 01010-$ ,  $B_3 = -00110, 00101-$ ,  $B_4 = -00011-$ . The union of these sets together with the allzero codeword yields the same basis  $B$ , as calculated by the coloring algorithm in Example 3.

## References

1. G. Forney. Coset codes - part ii: Binary lattices and related codes. *IEEE Trans. Inform. Theory*, 34:1152–1187, 1988.
2. G. Graetzer. *Universal Algebra*. D. van Nostrand Company, Inc., London/Toronto/Melbourne, 1968.
3. F. Kschischang and V. Sorokine. On the trellis structure of block codes. *IEEE Trans. Inform. Theory*, 41:1924–1937, 1995.
4. J. Maucher. On the theory of rectangular codes. *Ph.D Thesis, Department of Information Technology, University of Ulm*, 1999.
5. D. Muder. Minimal trellises for block codes. *IEEE Trans. Inform. Theory*, 34:1049–1053, 1988.
6. V. Sidorenko, G. Makarian, and B. Honary. Minimal trellis design for linear codes based on the shannon product. *IEEE Trans. Inform. Theory*, 42:2048–2053, 1996.
7. V. Sidorenko, J. Maucher, and M. Bossert. Rectangular codes and rectangular algebra. *Proc. of 13.th AAECC Symposium, LNCS*, Nov. 1999.
8. V. Sidorenko, J. Maucher, and M. Bossert. On the theory of rectangular codes. *Proc. of 6.th International Workshop on Algebraic and Combinatorial Coding Theory*, Sept. 1998.
9. V. Sidorenko, J. Maucher, M. Bossert, and R. Lucas. Rectangular codes in iterative decoding. *Proc. of ITG Fachtagung*, Jan. 2000.

# Graph Decoding of Array Error-Correcting Codes

Patrick G. Farrell<sup>1</sup>, Seyed H. Razavi<sup>2</sup>

<sup>1</sup> Lancaster University, UK  
[P.G.Farrell@lancaster.ac.uk](mailto:P.G.Farrell@lancaster.ac.uk)

<sup>2</sup> Curtin University of Technology, Perth, Australia

**Abstract.** The motivation for this paper is to report on concepts and results arising from the continuation of a recent study [1] of graph decoding techniques for block error-control (detection and correction) codes. The representation of codes by means of graphs, and the corresponding graph-based decoding algorithms, are described briefly. Results on the performance of graph decoding methods for block codes of the array and generalised array type will be presented, confirming the illustrative examples given in [1]. The main novel result is that the (7,4) Generalised Array Code, equivalent to the (7,4) Hamming Code, which has a graph which contains cycles, can be successfully decoded by means of an iterated min-sum algorithm.

## 1. Introduction

Graph decoding, using soft-decision methods, is potentially capable of providing simpler decoder implementations than other techniques. This applies particularly to list decoders, serial and parallel coding schemes, and iterative (eg, turbo) decoding algorithms. In these cases it is necessary to pass soft-decision information between the various stages of decoding, and graph-based decoding algorithms (eg, max- or min-sum and sum-product) are ideally suited for this purpose. In addition, they provide optimum symbol-by-symbol decoding as well as maximum and near-maximum likelihood codeword decoding.

Representation of a code by means of a graph was originally proposed by Tanner in 1981 [2]. Tanner used an iterative decoding algorithm previously discovered by Gallager in 1962 [3], and applied by him to the decoding of low-density parity-check codes. However, the power of this combination of a graphical representation and an iterative decoding algorithm was not fully realised until the work of Wiberg, Loeliger and Kotter in 1995 and 1996 [4,5]. This has led to a flurry of research into graph decoding, which is partly summarised in [1] and a paper by Forney [6]. It is interesting to note that very recent results on graph-based decoding have led to the creation of analog decoders, which outperform digital decoders by two orders of magnitude in speed and/or power consumption [8].

It turns out that array code and generalised array code (GAC) constructions [7] for block codes (and almost all optimum and well-known block codes can be constructed

in this way) facilitate and simplify the graph decoding of block codes. There are two main reasons for this. Firstly, the coset decomposition structure of a GAC leads to a corresponding decomposition of the Tanner graph into several disjoint sub-graphs, which in many cases do not contain cycles. This is important because almost all interesting codes have Tanner graphs with cycles, which then makes it difficult to apply a graph-based decoding algorithm. This coset decomposition was demonstrated in [1], with illustrative examples. Results using a simplified version of the max-sum algorithm are given below. Secondly, array codes and GACs can relatively easily be characterised by sectionalised Tanner and state [1,6] graphs, in which each node represents more than one codeword symbol. This leads to a simplified graph, with fewer or no cycles, thus in turn simplifying the decoding algorithm. Some results were given in [1], and will also be the subject of a future paper.

Even with coset decomposition and/or sectionalisation, some code graphs or sub-graphs will turn out to contain cycles. It is therefore of interest to explore ways of applying the max-sum and other “belief propagation” algorithms [4,5] to graphs with cycles. The results of such a study are reported below for the particular case of the binary Hamming code with block length  $n = 7$ ,  $k = 4$  information bits and minimum distance  $d = 3$ , formulated as a GAC structure, using an iterative and simplified max-sum algorithm. This algorithm is not the one introduced in [1], which on further investigation turned out to be faulty. The present algorithm, however, can correct single hard errors in any position in the (7,4) codewords; simulation results for its performance under additive white Gaussian noise conditions (soft errors) also will be reported in a future paper.

## 2. The Tanner Graph and Max-Sum Algorithm

The Tanner graph [2] of a binary, block, error-correcting code is a bipartite graph specifying the parity check relationships of the code. Each position (bit) of a codeword in the code is represented by one of a first set of nodes in the graph. All the position nodes in a parity relationship are joined by edges to one of a second set of nodes called parity nodes. There are  $n$  position nodes and  $n-k$  parity nodes in the Tanner graph of an  $(n,k,d)$  code.

The Tanner graphs of the (7,3,3) Array code and the (7,4,3) GAC are shown in Figures 1 and 2. The (7,3,3) code is constructed using the (2,1,2) row code and the (4,3,2) column code, with the check-on-checks bit then removed [7]. The parity relations are therefore between positions 1 and 2, 3 and 4, 5 and 6, and 1,3,5 and 7; as Figure 1 confirms. The graph does not contain cycles (ie, is a tree). This construction is also used as the basic Array code for the (7,4,3) code, but in addition a binary codeword (000 or 111) from the (3,1,3) repetition code is added to the bits in positions 2, 4 and 6. This then creates the (7,4,3) GAC code [7]. The generator matrix of this code is:

$$\mathbf{G} = \begin{pmatrix} 1100001 \\ 0011001 \\ 0000111 \\ 0101010 \end{pmatrix}$$

which leads to the parity check matrix:

$$\mathbf{H} = \begin{pmatrix} 1111000 \\ 0011110 \\ 1010101 \end{pmatrix}$$

as illustrated in Figure 2. It can be seen that this graph contains cycles.

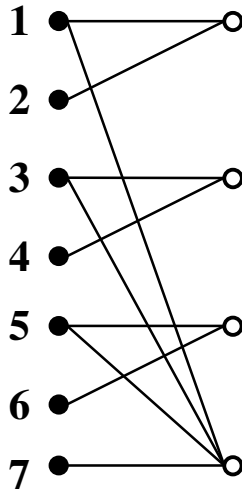
Details of the max-sum decoding algorithm are given in [1,4,5,6]. Briefly, the Tanner graph of a code is realised as a set of position nodes linked by edges to adders which implement the parity nodes. Assuming that the graph does not contain cycles, the algorithm starts simultaneously from all the outer nodes, moves towards the root of the tree and then propagates back towards the outer nodes. The initial log-likelihood weights  $w(0) = \log\{\text{prob}(y/x = 0)\}$  and  $w(1) = \log\{\text{prob}(y/x = 1)\}$  of each received signal element  $y$  (representing the transmitted bit  $x$  plus noise and other possible impairments) are allocated to the corresponding nodes. As the algorithm propagates through the graph, the weights are processed as follows:

- at a node, the outgoing weights are the sums of the corresponding incoming and initial weights, the final weights at an inner node are the sums of the incoming and outgoing weights on any edge attached to the node, and the final weight of an outer node is the sum of the incoming and initial weights;
- an adder, the outgoing weights are the maxima of the sums of all the corresponding incoming weights, over the set of all possible incoming bit configurations.

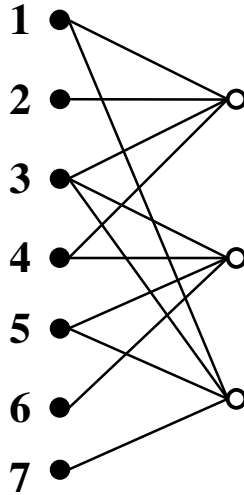
In the binary case, only the difference in the weights matters during the algorithm computations, so only a single metric (which now may be negative as well as positive) given by  $w(0) - w(1)$ , is used in the processing, as follows:

- at a node, the metric (difference) is the algebraic sum of the corresponding node metrics, combined as before;
- at an adder, the outgoing metric has magnitude given by the minimum value of the incoming metrics, and sign given by the sign of the product of all the incoming metrics; it is zero if any one of the incoming metrics is zero.

Thus, with this simplification, the max-sum algorithm becomes a min-sum algorithm [1,6]. The final result of using either decoding algorithm is the same, of course, except for the normalisation inherent in the simplified min-sum algorithm, and the same implications follow.



**Fig. 1.** Tanner graph of the  $(7,3,3)$  Array code



**Fig. 2.** Tanner graph of the  $(7,4,3)$  GAC

### 3. Decoding the (7,3,3) Array Code

The realisation of the graph of the code, given in Figure 1, is shown in Figure 3. If the initial metric at nodes 1,2, 4-7 is 4, corresponding to high confidence zeros, and is -2 at node 3, corresponding to a lowish confidence one, then applying the min-sum algorithm leads to final metrics of 10 for nodes 1,2,5,6 and 6 for the remaining nodes. The single error in node 3 is corrected, as expected, and the result also shows that positions 4 and 7 in the codeword are more affected by the error in position 3 than the other positions, as indicated by the lower confidence metric values.

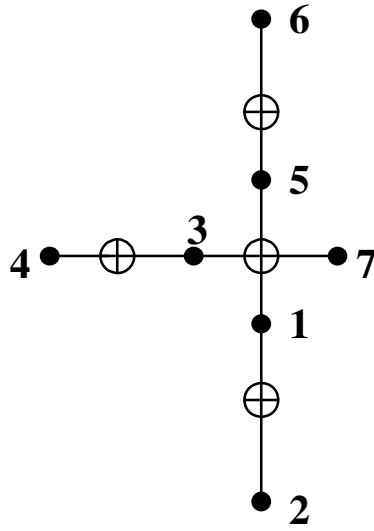
### 4. Decoding the (7,4,3) GAC

As Figure 2 shows, the Tanner graph for the (7,4,3) code is not a tree, because it contains cycles or loops. Regardless of the way in which the code is constructed, its corresponding Tanner graph will always contain cycles. In order to use the min-sum decoding algorithm with this code, it is therefore necessary to find ways in which to avoid or overcome the problem of the presence of the cycles.

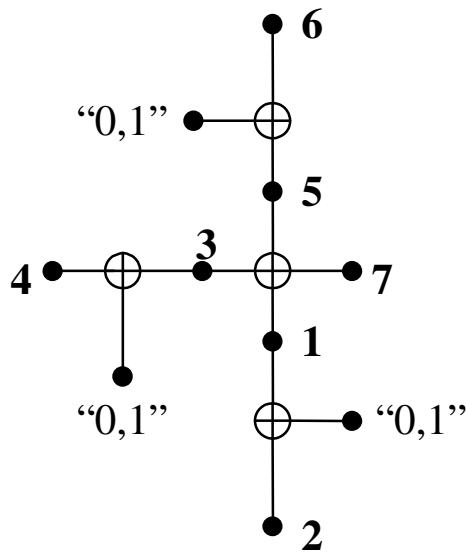
#### 4.1 Using the Coset Graph

A first way of dealing with the problem is to take advantage of the GAC form of the code, as described above. This structure means that the codewords in the code can be classified into two cosets: the first coset comprises the codewords of the (7,3,3) Array code, and the second coset consists of the same codewords but with the 2<sup>nd</sup>, 4<sup>th</sup> and 6<sup>th</sup> positions in the codewords inverted (complemented). The first coset is obtained when the 000 codeword in the (3,1,3) repetition code is added to positions 2, 4, and 6 in the basic (7,3,3) codeword (see Section 2 above); adding the 111 codeword then creates the second coset. Thus the (7,4,3) code can be represented by a pair of graphs, one for each coset. The graph corresponding to the first coset is identical to the graph for the (7,3,3) code, and the second graph is the same except that the bits in positions 2, 4, and 6 are inverted (which is equivalent to multiplying their metric values by -1) for the decoding process. The (7,4,3) GAC graph may therefore be drawn as in Figure 4; the min-sum algorithm is applied first with positions 2, 4, and 6 non-inverted, and then secondly with them inverted. In practice these two passes can be done serially or in parallel, as convenient. The results of each pass are then appropriately combined to obtain the final metrics for each bit.

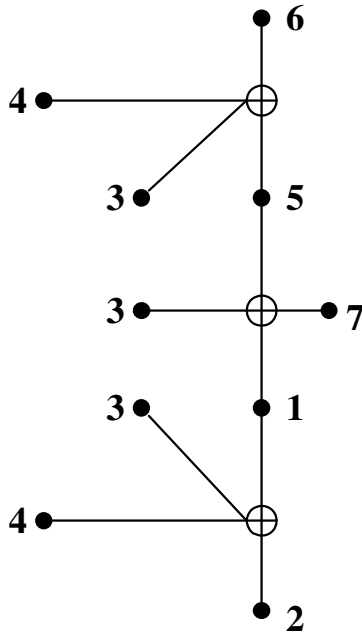
For example, let a set of initial metrics be 4, -4, 4, 4, 4, -4, and 4 in bit positions 1, 2, .....,7 respectively. This corresponds to receiving the codeword 0101010 with a hard error in position 4. After combining the results of the first and second decoding passes, the final node metrics are 4, -4, 4, -4, 4, -4, 4. The error in position 4 is corrected, and all bits have the same final confidence value.



**Fig. 3.** Realisation of the (7,3,3) Array code graph



**Fig. 4.** (7,4,3) GAC coset graph



**Fig. 5.** Split graph for the (7,4,3) GAC

## 4.2 Using a Split Graph

The second way of dealing with the problem is to modify the graph so as to remove the cycles, and then to repeatedly apply the min-sum algorithm until the final metrics converge satisfactorily. The modification consists of splitting an appropriate set of nodes so as to remove the cycles and create a tree graph. In general there are several ways of doing this. Figure 5 illustrates one way of modifying the graph of the (7,4,3) code shown in Figure 2, by splitting node 3 into three nodes, and node 4 into two nodes. In effect, this has replicated bit positions 3 and 4 in the codewords of the code, which would give an unfair weight to the initial metrics of nodes 3 and 4. Therefore the initial metric on each of the three nodes corresponding to position 3 should be only one third of the original value, and one half of the initial value on each of the two nodes corresponding to position 4. With these modifications, the min-sum algorithm can be applied repeatedly, in a number of iterations.

For example, let a set of initial metrics be 4, -4, -4, -4, 4, -4, 4 in bit positions 1, 2, ..., 7 respectively. This corresponds to receiving the codeword 0101010 with a hard error in position 3. After splitting nodes 3 and 4, and reducing their initial metrics

accordingly, the set of initial values becomes 4, -4, -1.33, -1.33, -1.33, -2, -2, 4, -4, 4. The final metrics after two iterations of the min-sum algorithm are as follows:

- 1.33, -2.67, 2.67, -1.33, 1.33, -2.67, 2.67
- 2.89, -3.34, 6, -3.11, 2.89, -3.34, 3.56

Note that the error in position 3 has been corrected, and that the metrics of the other positions are converging back to their original high confidence values after initial falls. Errors in other positions are similarly correctable, but may require up to four iterations.

## 5. Conclusions

Two methods which permit the use of the min-sum (or max-sum) graph decoding algorithm for block codes with Tanner graphs containing cycles have been described.

The coset graph method becomes computationally complex if the code has a large number of cosets, as many codes of practical interest do. It is therefore relevant to consider methods for limiting the number of cosets which have to be searched [9], though then the final metric values may not be very accurate. The method also depends on the basic coset having a cycle-free graph, and again this will not necessarily be so in many cases of interest. One way to increase the number of cycle-free coset graphs is to create suitable “sectionalised” Tanner graphs, with nodes representing more than one codeword position, and “adders” now operating on bit sequences rather than individual bits. State graphs [4,5,6] also seem very promising.

The split graph method warrants much further investigation. It is not clear, for example, when iteration should stop and how accurate the final metric values are after the last iteration. Computer simulations to determine the performance of the method for a range of codes and error conditions are also required. Is there an optimum way to split the cycle graph? Can state, coset and split graph techniques be combined in some way, to derive more effective soft decoding algorithms? These and other interesting questions will be addressed in subsequent papers.

## References

- [1] P.G. Farrell: Graph Decoding of Error-Control Codes; 5<sup>th</sup> Int. Symposium on DSP for Communication Systems, Scarborough, Perth, Australia, 1-4 February, 1999.
- [2] R.M. Tanner: A Recursive Approach to Low-Complexity Codes; IEEE Trans Info Theory, Vol IT-27, No 5, pp533-547, Sept 1981.
- [3] R.G. Gallager: Low-Density Parity-Check Codes; IRE Trans Info Theory, Vol IT-8, No 1, pp 21-28, Jan 1962.
- [4] N. Wiberg, H.-A. Loeliger & R. Kotter: Codes and Iterative Decoding on General Graphs; Euro Trans Telecom, Vol 6, pp513-526, SEPT 1995.
- [5] N. Wiberg: Codes and Decoding on General Graphs; PhD Dissertation, Linköping University, Sweden, April 1996.

- [6] G.D. Forney: On Iterative Decoding and the Two-Way Algorithm; Int Symp on Turbo Codes, Brest, France, Sept 3-5, 1997.
- [7] P.G. Farrell: On Generalised Array Codes; in Communications Coding and Signal Processing, Eds B. Honary, M. Darnell and P.G. Farrell, Research Studies Press, 1997.
- [8] H.-A. Loeliger, F. Tarkoy, F. Lustenberger & M. Helfenstein: Decoding in Analog VLSI; IEEE Comms Mag, April 1999, pp 99-101.
- [9] I. Martin & B. Honary: Two-Stage Trellis Decoding of the Nordstrom-Robinson Code Based on the Twisted Squaring Construction, submitted to IEE Proceedings - Communications.

# Catastrophicity Test for Time-Varying Convolutional Encoders

Conor O'Donoghue<sup>1</sup> and Cyril Burkley<sup>2</sup>

<sup>1</sup> Silicon & Software Systems, South County Business Park,  
Leopardstown, Co. Dublin, Ireland

conoro@s3group.com

<sup>2</sup> Dept. of Electronic Engineering, University of Limerick,  
Limerick, Ireland

cyril.burkley@ul.ie

**Abstract.** A new catastrophicity test for convolutional encoders whose rate and generator polynomials vary with time is presented. Based on this test computationally efficient algorithm to determine whether or not a time-varying convolutional encoder is catastrophic is derived. This algorithm is shown to be simpler than the catastrophicity test proposed by Balakirsky [1]. Furthermore, the algorithm can easily be generalised to rate  $k/n$  time-varying convolutional encoders.

## 1 Introduction

Let  $F_q[D]$  denote the ring of *polynomials* in the indeterminate  $D$  with elements  $a(D) = \sum_{i=0}^m a_i D^i$ ,  $m \geq 0$ , and  $a_i \in F_q$ , where  $F_q$  is some finite field with  $q$  elements and  $q$  is a prime power. An  $n$ -vector of polynomials,  $\mathbf{a}(D) = (a_1(D), a_2(D), \dots, a_n(D))$ , is an element of  $F_q[D]^n$ . The degree of  $\mathbf{a}(D)$  is defined as the maximum degree of its components

$$\deg \mathbf{a}(D) = \max_i \deg a_i(D) \quad (1)$$

A rate  $k/n$  fixed convolutional code  $\mathcal{C}$  may be generated by any  $F_q[D]$ -matrix

$$G(D) = \begin{pmatrix} g_{11}(D) & g_{1n}(D) & \mathbf{g}_1(D) \\ \vdots & \vdots & \vdots \\ g_{k1}(D) & g_{kn}(D) & \mathbf{g}_k(D) \end{pmatrix} = \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \quad (2)$$

whose rows span  $\mathcal{C}$ . The  $i$ th constraint length and the overall constraint length of  $G(D)$  are defined as  $v_i = \deg \mathbf{g}_i(D)$  and  $v = \sum_{i=1}^k v_i$ , respectively. The memory of  $G(D)$  is defined as  $v_m = \max_i v_i$ . Thus, we can write

$$G(D) = \sum_{i=0}^{v_m} G_i D^i \quad G_i \in F_q^{k \times n} \quad (3)$$

A convolutional encoder is said to be *catastrophic* if there exists some input sequence,  $\mathbf{u}(D)$ , with infinite Hamming weight which generates a code sequence,

$\mathbf{y}(D) = \mathbf{u}(D)G(D)$ , with finite Hamming weight. Such encoders are undesirable as a finite number of channel errors may give rise to an infinite number of errors in the decoded sequence. A necessary and sufficient condition for catastrophic convolutional encoders was first obtained by Massey and Sain [2] and generalised by Olsen [3]. Let  $\Delta_i(D)$  be the  $i$ th full size minor of  $G(D)$  and define

$$\delta(D) = \gcd \Delta_1(D), \dots, \Delta_L(D) \quad (4)$$

where  $L = \binom{n}{k}$ . Then  $G(D)$  is noncatastrophic if, and only if,  $\delta(D) = D^d$  for some  $d \geq 0$ .

A generator matrix,  $G(D)$ , for  $\mathcal{C}$  is *canonical* if its realisation in controller canonical form is minimal i.e. there are no encoders for  $\mathcal{C}$  requiring fewer memory elements. If  $G(D)$  has constraint lengths  $v_i$   $\frac{k}{1}$  the high order coefficient matrix,  $G_h = F_q^{k \times n}$ , is the matrix whose  $i$ th row consists of the coefficients of  $D^{v_i}$  in the  $i$ th row of  $G(D)$ . The following theorem, due to Forney [4,5], states when a generator matrix is canonical

**Theorem 1** Let  $G(D) = F_q[D]^{k \times n}$  be a generator matrix for some  $\mathcal{C}$  with overall constraint length  $v$ . Then the following statements are equivalent:

- (a)  $G(D)$  is a *canonical* generator matrix.
- (b) (i) The gcd of the  $k \times k$  minors of  $G(D)$  is 1 and  
(ii) their greatest degree is  $v$ .
- (c) (i)  $G(D)$  is noncatastrophic and  
(ii)  $G_0$  and  $G_h$  have full rank. □

The constraint lengths of any canonical generator matrix are invariants of the code  $\mathcal{C}$  and are called the *Kronecker indices* of  $\mathcal{C}$ , and denoted by  $\nu_i$   $\frac{k}{1}$ . The sum  $\nu = \sum_{i=1}^k \nu_i$  is simply referred to as the Kronecker index and is a measure of the complexity of  $\mathcal{C}$ .

## 2 Time-Varying Convolutional Codes

Consider  $N$  convolutional codes of rates  $1/n_1, \dots, 1/n_N$  and constraint length at most  $\mu$  defined by the generator polynomials

$$G^i(D) = [g_0^i(D) \ g_1^i(D) \ \dots \ g_{n_i}^i(D)] \quad (1 \leq i \leq N) \quad (5)$$

We define a *selection function*  $\varphi$  such that  $\varphi(t) = 1, 2, \dots, N$  and

$$\varphi(t + iT) = \varphi(t) - t, i \quad (6)$$

As  $\varphi(t)$  is periodic it is completely specified by the  $T$ -tuple  $\boldsymbol{\varphi} = (\varphi(0), \dots, \varphi(T-1))$ . Consider an information sequence  $u_0, u_1, \dots$ . Then the time-varying convolutional encoder output at time  $t$  is given by

$$\mathbf{y}_t = \left( \sum_{i=0}^{\mu} u_{t-i} G_i^{\varphi(t)} \right) \quad \mathbf{y}_t = F_q^{n_{\varphi(t)}} \quad (7)$$

The encoder may be realised with a single shift register of length  $\mu$  and time-varying connection vectors. In order to maximise utilisation of the available memory it is commonly assumed [6] that

$$G_0^i = 0 \quad \text{and} \quad G_\mu^i = 0 \quad 0 - i < T \quad (8)$$

All the best time-varying codes reported in [7] and [8] satisfy this condition. Further justification for this assumption is provided at the end of this section.

The rate of the time-varying code is given by  $R_\varphi = 1/n_\varphi$  where

$$n_\varphi = \frac{n_\varphi(0) + n_\varphi(1) + \dots + n_\varphi(T-1)}{T} \quad (9)$$

Note that  $n_\varphi$  may not be an integer. Nevertheless, writing  $R_\varphi$  in this form emphasises the fact that the time-varying code is described by a trellis corresponding to a rate  $1/n$  fixed code but with branch labels varying on successive trellis sections.

It is well known [7] that any periodic time-varying convolutional code with memory  $\mu$  and period  $T$  is equivalent to a fixed rate  $T/n_\varphi T$  code with generator matrix

$$G(D) = \left( \begin{array}{c} v_m \\ j=0 \end{array} G_j D^j \quad G_j - F_q^{T-n_\varphi T} \right) \quad (10)$$

where  $v_m = -\mu/T$ —and

$$G_j = \begin{pmatrix} G_{jT}^{\varphi(0)} & G_{jT+1}^{\varphi(1)} & \dots & G_{(j+1)T-1}^{\varphi(T-1)} \\ G_{jT-1}^{\varphi(0)} & G_{jT}^{\varphi(1)} & \dots & G_{(j+1)T-2}^{\varphi(T-1)} \\ \vdots & \vdots & \ddots & \vdots \\ G_{(j-1)T+1}^{\varphi(0)} & G_{(j-1)T+2}^{\varphi(1)} & \dots & G_{jT}^{\varphi(T-1)} \end{pmatrix} \quad (11)$$

where, by convention,  $G_j^i = 0$  for  $j < 0$  and for  $j > \mu$ . Letting  $\kappa = v_m T - \mu$ , the constraint lengths of  $G(D)$  can be shown to be

$$v_i = \begin{cases} v_m - 1 & 1 - i - \kappa \\ v_m & \kappa < i - T \end{cases} \quad (12)$$

The overall constraint length is, therefore,  $v = \mu$ . This equivalence of time-varying and fixed codes means that one can think of a time-varying code with period  $T$  and constraint length  $\mu$  as a special case of a rate  $T/n_\varphi T$  code which can be decoded with a simpler  $T$ -stage decoder with two additions and a binary comparison instead of a single stage decoder with  $2^T$  additions and a  $2^T$ -ary comparison for each state.

It is worth noting that using  $\varphi(t) = \varphi(t + \tau)$ ,  $\tau \in \mathbb{Z}$ , instead of  $\varphi(t)$  results in essentially the same time-varying code. However, the corresponding fixed code will, in general, be different. Therefore, for every periodic time-varying code there are  $T$  equivalent fixed convolutional codes. Now, consider the case where

$G_0^r = 0$  for some  $1 \leq r \leq N$ . Choose  $\tau$  such that  $\varphi^- = (\dots, r)$ . The generator matrix for the equivalent fixed convolutional code has a low order coefficient matrix

$$G_0 = \begin{pmatrix} G_0^{\varphi'(0)} & G_1^{\varphi'(1)} & \dots & G_{T-1}^{\varphi'(T-1)} \\ 0 & G_0^{\varphi'(1)} & \dots & G_{T-2}^{\varphi'(T-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & G_0^{\varphi'(T-1)} \end{pmatrix} \quad (13)$$

But  $G_0^{\varphi'(T-1)} = G_0^r = 0$  and hence the last row of  $G_0$  is zero. As a result the encoder has non-zero delay and therefore  $\nu < \mu$ . A similar argument may be used to show that, if there is some  $r$  for which  $G_\mu^r = 0$ , then the generator matrix of at least one of the equivalent fixed codes will have constraint length  $\nu < \mu$ . Since  $\nu = v$  it follows that  $\nu < \mu$ . Therefore all time-varying encoders not satisfying (8) are equivalent to fixed codes with Kronecker index  $\nu < \mu$  and therefore will have poorer distance properties. Consequently (8) is assumed throughout the remainder of this paper.

### 3 Catastrophicity Test

A time-varying encoder can be tested for catastrophicity by computing the gcd of the full size minors of  $G(D)$ , the generator matrix of the equivalent fixed code.

**Example 1** Consider the time-varying code defined by  $\varphi = (0, 1)$  and the generator matrices

$$G^0(D) = [1 + D^3 \quad 1 + D + D^2] \quad G^1(D) = [1 + D \quad 1 + D^3]$$

The equivalent rate 2/4 fixed convolutional code has generator matrix

$$G(D) = \begin{bmatrix} 1 & 1 + D & 1 & D \\ D^2 & D & 1 & 1 \end{bmatrix}$$

$\delta(D) = 1$  and hence the encoder is noncatastrophic.

**Example 2** A time-varying code is defined by  $\varphi = (0, 1, 2)$  and the generator matrices

$$G^0(D) = [1 + D \quad 1 + D + D^2] \quad G^1(D) = [1 \quad 1 + D + D^2] \\ G^2(D) = [D \quad 1 + D^2]$$

The generator matrix for the equivalent rate 3/6 fixed convolutional code is

$$G(D) = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & D & 1 & 1 & 1 & 0 \\ D & D & 0 & D & 0 & 1 \end{pmatrix}$$

In this case  $\delta(D) = 1 + D$  and therefore the encoder is catastrophic.

These examples demonstrate that catastrophicity is not inherited from the encoders  $G^i(D)$ ,  $1 - i - N$ . In Example 1 both  $G^0(D)$  and  $G^1(D)$  are catastrophic but the time-varying encoder is noncatastrophic. The reverse is the case in Example 2. Note also that if  $\varphi = (0, 2, 1)$  is used in Example 2 the resulting time-varying code is noncatastrophic.

In general, the gcd method is not suitable for use in computer searches for good codes as it is not computationally efficient and the complexity grows exponentially with  $T$ . For example, a time-varying code with period  $T = 4$  and  $n_\varphi = 3$  requires computing  $495 \cdot 4 - 4$  minors and then finding their greatest common divisor. As an alternative, Balakirsky [1] derived a necessary and sufficient condition for catastrophicity based on the properties of autoregressive filters. However, it is not clear that the computational complexity of Balakirsky's test is significantly lower than that of the gcd method. In this section we present a new catastrophicity test for time-varying convolutional codes. A fast algorithm implementing the test is derived in Section 4.

**Lemma 1** A periodic time-varying encoder with generator matrices  $G^i(D)$ ,  $1 - i - N$ , and selection function  $\varphi(t)$  is noncatastrophic if, and only if,  $G(D)$ , the generator matrix for an equivalent fixed code, is canonical.  $\square$

*Proof.* Let  $\mathcal{C}$  be the rate  $T/n_\varphi T$  code generated by  $G(D)$ . The low- and high-order coefficient matrices of  $G(D)$  are given by

$$G_0 = \begin{pmatrix} G_0^{\varphi(0)} & \dots & G_{T-1}^{\varphi(T-1)} \\ 0 & \ddots & \vdots \\ 0 & 0 & G_0^{\varphi(T-1)} \end{pmatrix} \quad G_h = \begin{bmatrix} 0 & I_\kappa \\ I_{T-\kappa} & 0 \end{bmatrix} \begin{pmatrix} G_\mu^{\varphi(0)} & 0 & 0 \\ \vdots & \ddots & 0 \\ G_{\mu-T+1}^{\varphi(0)} & \dots & G_\mu^{\varphi(T-1)} \end{pmatrix} \quad (14)$$

where  $\kappa = T - \mu/T - \mu$ . Since  $G_0^{\varphi(i)} = 0$  and  $G_\mu^{\varphi(i)} = 0$  for all  $1 - i < T$  it follows that both  $G_0$  and  $G_h$  have full rank. Therefore, by statement (c) of Theorem 1, the time-varying encoder is canonical if, and only if, it is noncatastrophic.  $\square$

Thus we may test a time-varying encoder for catastrophicity using the following canonicity test for rate  $k/n$  fixed convolutional encoders [9, Theorem 6].

**Theorem 2** Let  $G(D) = F_q^{k-n}$  be any generator matrix with overall constraint length  $v$  and memory  $v_m$ . Then  $G(D)$  is canonical if, and only if,

$$\text{rank } \mathcal{H}_-[v] = (k + 1)v \quad (15)$$

where  $\mathcal{H}_-[\ell]$  is the  $(\ell + v_m)k - \ell n$  matrix

$$\mathcal{H}_-[\ell] = \begin{pmatrix} G_0 & 0 \\ \vdots & \ddots \\ G_{v_m} & G_0 \\ & \ddots & \vdots \\ 0 & G_{v_m} \end{pmatrix} \quad (16)$$

$\square$

With a view to reducing the number of computations required to determine the rank of  $\mathcal{H}_{-}[v]$  we will analyse the rank properties of this matrix in more detail. To do this we will use the following lemma due to Forney [4].

**Lemma 2** Let  $\mathcal{C}$  be any rate  $k/n$  fixed convolutional code with Kronecker indices  $-\nu_i - \frac{k}{1}$  and let  $\mathcal{C}_\ell$  denote set of polynomial codewords in  $\mathcal{C}$  with degree strictly less than  $\ell$ .

$$\mathcal{C}_\ell := \{\mathbf{y}(D) \in \mathcal{C} \mid \deg \mathbf{y}(D) < \ell, \det \mathbf{y}(D) \neq 0\} \quad (17)$$

Then  $\mathcal{C}_\ell$  is a subspace of  $\mathcal{C}$  over  $\mathbb{F}_q$  with dimension

$$\dim_{\mathbb{F}_q} \mathcal{C}_\ell = k\ell - \nu + \sum_{i: \nu_i \geq \ell} (\nu_i - \ell) \quad (18)$$

□

The rank of the matrix  $\mathcal{H}_{-}[\ell]$  is given by the following theorem.

**Theorem 3** Let  $\mathcal{C}$  be a rate  $k/n$  fixed convolutional code and let  $G(D)$  be any generator matrix for  $\mathcal{C}$  with constraint lengths  $-\nu_i - \frac{k}{1}$ . Then

$$\text{rank } \mathcal{H}_{-}[\ell] = k\ell + \nu - \sum_{i: \nu_i^\perp \geq \ell} (\nu_i^\perp - \ell) \quad (19)$$

where  $-\nu_i^\perp - \frac{n-k}{1}$  are the Kronecker indices of  $\mathcal{C}^\perp$ . □

*Proof.* Let  $H(D)$  be any canonical polynomial generator matrix for  $\mathcal{C}^\perp$ . Since  $\mathcal{C}$  and  $\mathcal{C}^\perp$  are dual subspaces of  $\mathbb{F}_q[D]^n$  it follows that

$$H(D)G^\top(D) = 0 \quad (20)$$

where the dash denotes matrix transposition. Equivalently we can write  $H\bar{G} = 0$  where  $H$  and  $\bar{G}$  are the semi-infinite block matrices

$$H = \begin{pmatrix} H_0 & H_1 & H_2 & H_3 & \dots \\ 0 & H_0 & H_1 & H_2 & \dots \\ 0 & 0 & H_0 & H_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad \bar{G} = \begin{pmatrix} G_0^- & G_1^- & G_2^- & G_3^- & \dots \\ 0 & G_0^- & G_1^- & G_2^- & \dots \\ 0 & 0 & G_0^- & G_1^- & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad (21)$$

Now let  $\bar{G}[\ell]$  denote the matrix consisting of the first  $\ell$  block rows and  $\ell + v_m$  block columns of  $\bar{G}$

$$\bar{G}[\ell] = \begin{pmatrix} G_0^- & G_1^- & \dots & G_{v_m}^- & 0 & 0 & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ 0 & 0 & G_0^- & G_1^- & \dots & G_{v_m}^- & \dots \end{pmatrix} \quad (22)$$

The kernel of  $\bar{G}[\ell]$  is spanned by those rows of  $H$  that are zero in all but the first  $n\ell$  columns. Since  $H(D)$  is a canonical polynomial generator matrix these

rows are also a basis for  $\mathcal{C}_\ell^-$ , the set of polynomial codewords in  $\mathcal{C}^-$  with degree less than  $\ell$ . Therefore  $\ker \bar{G}[\ell] = \mathcal{C}_\ell^-$  and hence from Lemma 2

$$\dim \ker \bar{G}[\ell] = (n - k)\ell - \nu + \sum_{i: \nu_i^- \geq \ell} (\nu_i^- - \ell) \quad (23)$$

We may also write  $\dim \ker \bar{G}[\ell] = n\ell - \text{rank } \bar{G}[\ell]$ . Substituting into (23) and re-arranging terms yields

$$\text{rank } \bar{G}[\ell] = k\ell + \nu - \sum_{i: \nu_i^- \geq \ell} (\nu_i^- - \ell) \quad (24)$$

But inspection of (22) reveals that  $\bar{G}[\ell]$  is the transpose of  $\mathcal{H}_-[\ell]$ . Therefore

$$\text{rank } \bar{G}[\ell] = \text{rank } \mathcal{H}_-[\ell] \quad (25)$$

Substituting the expression for  $\text{rank } \bar{G}[\ell]$  given by (24) yields the desired result.  $\square$

Combining Lemma 1 and Theorems 2 and 3 we obtain our main result.

**Theorem 4** A periodic time-varying encoder with generator matrices  $G^i(D)$ ,  $1 \leq i \leq N$ , constraint length  $\mu$ , and selection function  $\varphi(t)$  is noncatastrophic if, and only if,

$$\text{rank } \mathcal{H}_-[\ell] = T\ell + \mu - \ell - \nu_m^- \quad (26)$$

where  $G(D)$  is the generator matrix of an equivalent rate  $T/n_\varphi T$  fixed code  $\mathcal{C}$  and  $\nu_m^-$  is the largest Kronecker index of  $\mathcal{C}^-$ .  $\square$

We note that  $\frac{\nu}{(n_\varphi - 1)T} = \nu_m^- - \nu$  and hence application of Theorem 4 may involve significantly fewer computations than computing the rank of  $\mathcal{H}_-[v]$ . In the next section we will use Theorem 4 as the basis of a fast algorithm to test a time-varying encoder for catastrophicity.

## 4 Fast Algorithm

A computationally efficient algorithm to implement the catastrophicity test of Theorem 4 can be obtained by exploiting the banded and block Toeplitz structure of the matrix  $\mathcal{H}_-[\ell]$ . We begin by permuting the columns of the matrices  $G_i$  such that  $G_0 = [G_{00} \ G_{01}]$  where  $G_{00} = F_q^{T-T}$  is a nonsingular upper triangular matrix. This is easy to do since there is at least one non-zero element in each of the block matrices on the diagonal of  $G_0$ . Multiplying the block rows of  $\mathcal{H}_-[\ell]$  by  $G_{00}^{-1}$  and re-ordering columns yields the matrix

$$\hat{\mathcal{H}}_-[\ell] = \left( \begin{array}{cc|cc} I & 0 & S_0 & 0 \\ R_1 & \ddots & S_1 & \ddots \\ \vdots & & \vdots & S_0 \\ R_{v_m} & I & S_{v_m} & S_1 \\ & \ddots & \vdots & \vdots \\ 0 & R_{v_m} & 0 & S_{v_m} \end{array} \right) \quad (27)$$

where  $[R_i \ S_i] = G_{00}^{-1} G_i$ ,  $R_i = F_q^{T-T}$ , and  $v_m = \mu/T$ . Using elementary row operations  $\hat{\mathcal{H}}_-[\ell]$  can be put in the form

$$\left( \begin{array}{cc|ccc} I & 0 & \bar{S}_0 & & 0 \\ & \ddots & \vdots & \ddots & \\ 0 & I & \bar{S}_{\ell-1} & \dots & \bar{S}_0 \\ \hline 0 & \dots & 0 & \bar{S}_\ell & \dots & \bar{S}_1 \\ \vdots & \vdots & \vdots & & \vdots & \\ 0 & \dots & 0 & \bar{S}_{\ell+v_m-1} & \dots & \bar{S}_{v_m} \end{array} \right) \quad (28)$$

where  $\bar{S}_i = F_q^{T-(n_\varphi-1)T}$ ,  $0 \leq i \leq \ell + v_m$ . It can be shown that the matrices  $\bar{S}_i$  are given by the recursion formula

$$\bar{S}_i = S_i + \sum_{j=0}^{i-1} R_{i-j} \bar{S}_j \quad (29)$$

Note that  $R_i = 0$ ,  $-i > v_m$  and hence the computation of  $\bar{S}_i$  requires at most  $v_m$  matrix products. Having computed the  $\bar{S}_i$  we form the matrix

$$W[\ell] = [W_1 \ W_2 \ \dots \ W_\ell] \quad (30)$$

where

$$W_i = \begin{pmatrix} \bar{S}_i \\ \vdots \\ \bar{S}_{i+v_m-1} \end{pmatrix} \quad (31)$$

From (28) it is easily seen that  $\text{rank } W[\ell] = \text{rank } \hat{\mathcal{H}}_-[\ell] - T\ell$ . Substituting the expression for  $\text{rank } \hat{\mathcal{H}}_-[\ell]$  given by (19) yields

$$\text{rank } W[\ell] = \nu - \sum_{i: \nu_i^\perp \geq \ell} (\nu_i^- - \ell) \quad (32)$$

and hence by Theorem 4 the encoder is non-catastrophic if, and only if,

$$\text{rank } W[\ell] = \mu - \ell - \nu_m^- \quad (33)$$

However, we have no a priori knowledge of  $\nu_m^-$ . This difficulty may be circumvented by noting that

$$\text{rank } W[\ell + 1] - \text{rank } W[\ell] = 0 \quad (34)$$

where the equality holds if, and only if,  $\ell = \nu_m^-$ . Hence we compute  $\text{rank } W[\ell]$  for  $\ell = 1, 2, \dots$  until either (i)  $\text{rank } W[\ell] = \mu$  or (ii)  $\text{rank } W[\ell] - \text{rank } W^{(\ell-1)} = 0$ . In both cases  $G(D)$  is non-catastrophic if, and only if,  $\text{rank } W[\ell] = \mu$ .

Finally, we may compute  $\text{rank } W[\ell]$ ,  $\ell = 1, 2, \dots$ , as follows. Using elementary column operations put  $W[\ell]$  in column echelon form, denoted here by  $W_c[\ell]$ . The rank of  $W_c[\ell]$  is determined by inspection.  $W_c^{[\ell+1]}$  is easily found from the augmented matrix  $[W_c[\ell] \ -W_{\ell+1}]$  where  $W_{\ell+1}$  is given by (31). The complete algorithm is summarised as follows:

```

Step 1  From the generator matrices  $G^i(D)$ ,  $1 - i - N$ , and the
        function  $\varphi(t)$ , construct the coefficient matrices  $-G_i \begin{smallmatrix} v_m \\ 0 \end{smallmatrix}$ .
Step 2  Compute  $G_{00}^{-1}$  and the matrices  $-R_i \begin{smallmatrix} v_m \\ 1 \end{smallmatrix}$  and  $-S_i \begin{smallmatrix} v_m \\ 0 \end{smallmatrix}$ .
Step 3  FOR  $\ell = 1$  TO  $\mu$ 
        Compute  $W_\ell$  using the recursion formula (29).
        Using elementary column operations obtain  $W_c[\ell]$ 
        from  $W_c[\ell - 1]$  and  $W_\ell$ .
        IF  $(\text{rank } W_c[\ell] - \text{rank } W_c[\ell - 1] = 0)$  OR  $(\text{rank } W_c[\ell] = \mu)$ 
            THEN GOTO END
        NEXT  $\ell$ 
Step 4  END. The encoder is noncatastrophic if  $\text{rank } W_c = \mu$ 

```

## 4.1 Computational Complexity

For simplicity we assume binary codes. Computing the matrices  $-\bar{S}_i$  requires  $O(\nu_m^- \mu T^2 n_\varphi)$  binary operations. Computation of the rank of  $W^{(\nu_m^\pm)}$  requires  $O(\nu_m^- \mu^2 T n_\varphi)$  binary operations. Typically  $\mu$  will be greater than  $T$  and consequently this latter step dominates the overall computational complexity of the algorithm.

## 5 Conclusions

We have presented a new algorithm for identifying rate  $1/n_\varphi$  catastrophic time-varying convolutional encoders. The algorithm requires no polynomial operations has a simple software implementation. The computational complexity is  $O(\nu_m^- T n_\varphi \mu^2)$  and is less complex than the algorithm proposed by Balakirsky. Furthermore, the algorithm presented here is easily generalised to rate  $k/n_\varphi$  time-varying codes.

## References

1. V.B. Balakirsky, "A necessary and sufficient condition for time-variant convolutional encoders to be noncatastrophic," *Lecture Notes in Computer Science*, No. 781, pp. 1-10, Springer-Verlag, 1993.
2. J.L. Massey and M.K. Sain, "Inverses of linear sequential circuits," *IEEE Trans. Computers*, Vol. C-17, No. 4, pp. 330-337, April 1968.
3. R.R. Olsen, "Note on feedforward inverses for linear sequential circuits," *IEEE Trans. Computers*, Vol. C-19, No. 12, pp. 1216-1221, Dec. 1970.
4. G.D. Forney, Jr., "Minimal bases of rational vector spaces, with applications to multivariable linear systems," *SIAM J. Control*, vol. 13, pp.493-520, May 1975.
5. R. Johannesson and Z.-X. Wan, "A linear algebra approach to convolutional encoders," *IEEE Trans. Inform. Theory*, vol. IT-39, No. 4, pp. 1219-1233, July 1993.
6. P.J. Lee, "There are many good time-varying convolutional codes," *IEEE Trans. Inform. Theory*, Vol. IT-35, No. 2, pp. 460-463, March 1989.
7. M. Mooser, "Some periodic convolutional codes better than any fixed code," *IEEE Trans. Inform. Theory*, Vol. IT-29, No. 5, pp. 750-751, Sept. 1983.
8. R. Palazzo, "A time-varying convolutional encoder better than the best time-invariant encoder," *IEEE Trans. Inform. Theory*, Vol IT-39, No.3, pp. 1109-1110, May 1993.
9. C. O'Donoghue, and C.J. Burkley, "Minimality and canonicity tests for rational generator matrices for convolutional codes," in *Proc. 1998 IEEE Information Theory Workshop*, pp. 112-114, Killarney, 22-26 June, 1998.

# Low Complexity Soft-Decision Sequential Decoding Using Hybrid Permutation for Reed-Solomon Codes

Min-seok Oh<sup>1</sup> and Peter Sweeney<sup>2</sup>

<sup>1</sup> CCSR, University of Surrey, Guildford, Surrey, GU2 5XH, UK  
m.oh@ee.surrey.ac.uk

<sup>2</sup> CCSR, University of Surrey, Guildford, Surrey, GU2 5XH, UK  
p.sweeney@ee.surrey.ac.uk

**Abstract.** We present a soft-decision decoding method for Reed-Solomon codes (RS codes) using both cyclic and squaring permutations. These permutations are used to provide a convenient sequence which is predicted to have relatively low complex error pattern with respect to a modified Fano sequential algorithm[1]. In order to preserve bit-level soft-decision values, each sequence of those permutation groups must keep equal weight distribution in symbol and bit level. Trellis construction is based on Wolf's method[2] and a binary systematic parity check matrix of RS codes is used for bit-level decoding[9]. In simulation results, it is shown that a hybrid of those two permutations can be used for low complexity decoding approaching maximum likelihood performance.

## 1 Introduction

Since Reed-Solomon codes[3] were introduced in 1960, many decoding methods have been developed. However, soft-decision decoding method could not be easily implemented because of complexity problem. Chase[5] and Forney[4] introduced interesting methods for soft-decision decoding of block codes. Their algorithms have tradeoffs between complexity and decoding performance for the application to RS codes. Later, some other approaches using trellis structure were developed and demonstrated some good results [6][7]. Despite such achievements, they did not fully use bit-level soft-decision information and could not solve complexity problem for long RS codes with a large field.

For the bit-level soft-decision decoding for RS codes, Vardy[8] presented a method using a union of codes being an interleaver of several binary BCH codes for representation of RS codes. Recently Oh and Sweeney presented another relatively simple method[9] which employs bit-level soft-decision information with low complexity. In this method a modified Fano algorithm was used with cyclic permutation of RS codes. In this work, although cyclic permutation contributes to considerable complexity reduction showing near-maximum likelihood performance(ML), it was not effective in decoding of a received sequence with widely distributed errors since a sequence given by a cyclic shift has a similar error pattern to the original one. In

order to deal with this kind of error pattern, squaring permutation[13][16] can be useful, since it can provide a different set of sequences compared with the cyclic permutation. However, squaring permutation is generally inferior to the cyclic permutation because of smaller size of permutation group.

In this paper, we present hybrid permutation which is a combination of cyclic and squaring permutations. For  $(n, k)$  RS codes over  $GF(2^m)$ , since cyclic and squaring permutation generate  $n$  and  $m$  different sequences respectively, the hybrid permutation provides  $m \cdot n$  different sequences from an original sequence. With this permutation, a sequential decoder can reduce complexity by performing a convenient sequence-first decoding.

Complexity characteristics of sequential decoding was well studied in [10][11][12]. In general the complexity depends on the error bits and error location in information block, it is reasonable to regard the sequence with the most reliable information block as the most convenient for sequential decoding. In this paper, we use a criterion which is represented by the sum of symbol confidences within information block of each sequence of permutation group. The symbol confidence is taken as the worst bit confidence within each symbol, since the decoding complexity will be affected by the worst one.

In section 2, we describe three permutation groups for RS codes: cyclic, squaring and hybrid permutation. Then, in section 3, we present *hybrid permutation sequential decoding* (HPSD) to achieve near-ML performance at reasonable complexity cost. Section 4 shows simulation results for HPSD in terms of error correcting performance and complexity.

## 2 Symbol Permutation of RS Codes

Permutation groups of RS codes provide many equivalent sequences which are useful for a low complexity decoding. When a received sequence contains some error, a permutation of the sequence may give a desirable effect that each permuted sequence can have different complexity due to changing the location of error bits. In this section we discuss three permutation techniques for RS codes: *cyclic*, *squaring* and *hybrid*.

### 2.1 Cyclic Permutation

We consider  $(n, k)$  Reed-Solomon codes over  $GF(2^m)$  and denote a code word denote  $c(x)$  as

$$c(x) = \sum_{i=0}^{n-1} c_i \cdot x^i \quad \text{for } c_i \in GF(2^m). \quad (1)$$

The elements of  $n$  cyclic permutation group  $T_c(c(x))$  are

$$\left( \sum_{i=0}^{n-1} c_i \cdot x^{i+\beta} \right) \bmod (x^n - 1) \text{ for } \beta \in (0, 1, 2, \dots, n-1). \quad (2)$$

By cyclic permutation of a code word,  $n$  different sequences are obtained and each sequence is also a code word in which the bits and confidences are also shifted with. Therefore a certain error pattern of a received sequence is changed by the cyclic permutation. A decoder can firstly choose the sequence with the most convenient error pattern among  $n$  possible sequences. A decoding method using this kind of permutation has been shown in[9].

## 2.2 Squaring Permutation

Squaring permutation is a technique using the property that although the squaring of a code word polynomial changes the position of symbols constituting the code word, the squared result is also a code word. For  $(n, k)$  Reed-Solomon codes over  $GF(2^m)$ , we can get  $m$ -different sequences which have the different error pattern.

A decoding approach using squaring permutation decoding was previously described[13] based on algebraic decoding method. However, we need further consideration for the application to a bit-level sequential decoding since bit-level soft decision information should be preserved through squaring operation.

In this paper, each symbol for RS code is represented on *normal basis*[14] which is defined as a set of linearly independent roots with the form  $\{\lambda_0, \lambda_1, \dots, \lambda_{m-1}\}$  for  $\lambda_i = \beta^{2^i}$ . On this basis, since the result of squaring of each symbol is represented by just a cyclic shift, the bit level soft-decision can be completely preserved through the squaring process. Perlis[15] has shown that a necessary and sufficient condition for a normal basis such as

$$tr(\beta) = \sum_{i=1}^{m-1} \beta^{2^i} = 1. \quad (3)$$

Table 1. shows a normal basis satisfying the above condition.

**Tabel 1.** Basis Representation for RS codes

Field	$GF(2^3)$	$GF(2^4)$	$GF(2^5)$
Polynomial basis	$\alpha^2 \alpha^1 \alpha^0$	$\alpha^3 \alpha^2 \alpha^1 \alpha^0$	$\alpha^4 \alpha^3 \alpha^2 \alpha^1 \alpha^0$
Normal basis	$\alpha^5 \alpha^6 \alpha^3$	$\alpha^9 \alpha^{12} \alpha^6 \alpha^3$	$\alpha^{17} \alpha^{24} \alpha^{12} \alpha^6 \alpha^3$

The elements of the  $m$  squaring permutation group  $T_s$  for  $s = 1, 2, \dots, m-1$ . are

$$\left( c(x) = \sum_{i=0}^{n-1} c_i x^i \right)^{2^s} = \sum_{i=0}^{n-1} c_i^{2^s} x^{(2^s \cdot i) \bmod n} \quad (4)$$

Let  $c_i$  denote  $c_i = \sum_{j=0}^{m-1} a_j \lambda_j$  for  $a_j \in GF(2)$  on the normal basis, the coefficient  $c_i^{2^s}$  is

expressed by  $c_i^{2^s} = \sum_{j=0}^{m-1} a_j \lambda_{(j+s) \bmod m}$ . Consequently since squaring permutation

using normal basis can preserve bit-level soft-decision values through the squaring operation, we can apply this technique to our bit-level sequential decoding. Moreover squaring of a symbol can be simply obtained by a bit cyclic shift within the symbol.

### 2.3 Hybrid Permutation

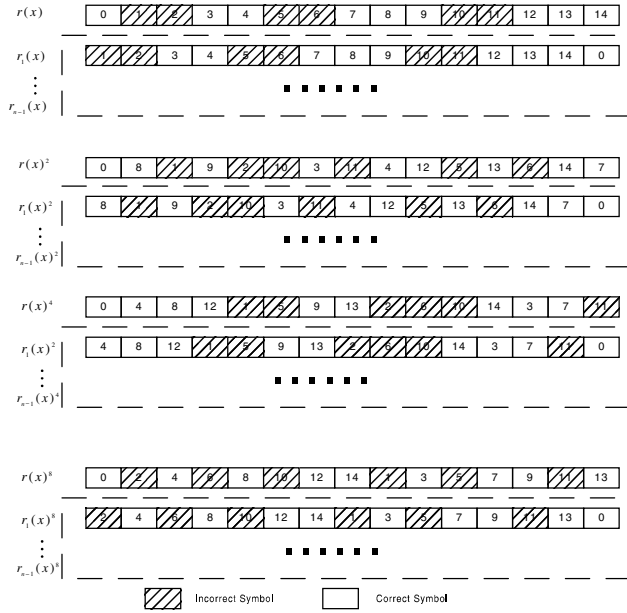
Hybrid permutation is the combination of the cyclic and squaring permutation. As we have examined in the previous section, since for  $(n, k)$  RS codes over  $GF(2^m)$  the  $m$  squaring sequences can have  $n$  cyclic permuted sequences: a total of  $n \times m$  permuted sequences can be obtained by combining the permutations. This means that an error pattern from a received sequence is also changed to other patterns with  $n \times m$  different complexity. For RS codes with a large field, the number of possible sequences increases. Table 1 shows the number of possible sequences by the hybrid-permutation.

**Table 1.** Possible Sequences by Hybrid Permutation

Field	8	16	32	64
Possible Sequences	21	60	155	378

Fig.1 shows error pattern changes by hybrid-permutation for (15,9) RS codes. In the figure, it is shown that four sequences are obtained from squaring permutation and then 15 cyclic sequences are produced each corresponding to each one of the squaring permutation group. Thus total 60 sequences can be obtained from a received sequence and each sequence has the same symbol and binary weights as the original sequence because of the use of normal basis. The different thing in each sequence is the order of listed symbols. Since the complexity of the sequential decoder depends on the locations of errors, it is expected that the complexity of decoding the original sequence can also be changed with the permutation.

Hybrid permutation is very attractive to design an efficient permutation sequential decoder, since the decoder can choose the most convenient sequence from a greater variety of sequences than either of the cyclic or squaring permutation individually. This permutation gives a solution for individual drawbacks of cyclic and squaring techniques which are used for permutation decoding. Widespread errors can be rearranged by the squaring permutation so that cyclic permutation can effectively manage the rearranged sequence. Therefore we can improve the complexity and decoding performance simultaneously. In particular, in the application of RS codes over a large Galois field, hybrid permutation will be very powerful in reducing complexity and improving decoding performance.



**Fig. 1.** Error Pattern Changes by Hybrid Permutation

### 3 Hybrid Permutation Sequential Decoder

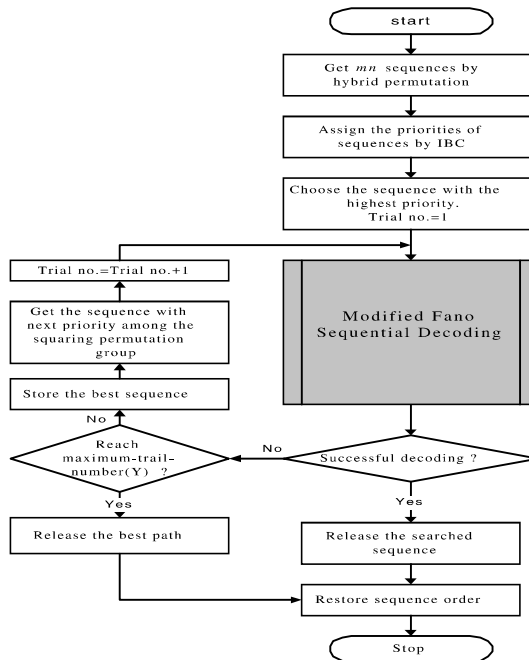
We present *hybrid permutation sequential decoder* (HPSD) which uses a modified Fano algorithm with hybrid permutation. The *modified Fano algorithm*[9] has two additional functions, which are *path update function* and *decision rule*, to the original Fano algorithm[8]. The path update function updates a searched path whenever its path metric is greater than current one. By the path updating, the decoder can release the best path. On the other hand, the *decision rule* is to qualify searched paths in the case that the decoder has searched for a wrong path as if it were the correct path. With those two additional functions, the modified Fano algorithm approaches maximum likelihood performance only if the decoder has tried the correct path at least once.

For the efficient operation of the sequential decoding, the decoding parameters are optimized as the most proper value for computational limit  $L$  and threshold spacing step,  $\Delta T$ . For the convenient sequence-first search, the decoder considers the convenience level as confidences with respect to the information part of a code word for the possible permuted sequences by hybrid-permutation. Those sequences are sorted by the sum of confidences of the information part and their priorities are assigned for decoding. The decoding procedure is explained as following:

- (i) Obtain  $m \times n$  candidates by cyclic and squaring permutation.
- (ii) Assign the decoding priority of the candidates in order of IBC(information block confidence) of each candidate. Set *trial number* to 1.

- (iii) Choose the sequence with the highest priority, which has the largest IBC.
- (iv) Decode the chosen sequence by using the modified Fano algorithm (MFA).
- (v) Check decoding result.
  - If the decoder has found a valid path satisfying the *decision rule*, release the path and restore its sequence order.
  - Otherwise store the best path which has been recorded so far by the *path update function*. Then go to the next step.
- (vi) Increase *trial number*.
  - If *trial number* is less than a given *maximum-trial-number*, choose the sequence with next priority and then go to step (iv).
  - Otherwise release the best path which has been recorded so far and then restore the permuted sequence with respect to the path.

Fig.2. shows the flow chart of the hybrid permutation sequential decoder.



**Fig.2.** Hybrid Permutation Sequential Decoder

## 4 Simulation Results

Simulation was carried out on BPSK system with 8-level soft decision values over Gaussian channel. Decoding performance was obtained for (7,3), (15,9) and (31,27) RS codes in terms of complexity and decoding error rate as a function of  $E_b/N_0$ . The complexity was measured by average computations per information bit and error correcting performance was calculated by bit error rate (BER) with respect to  $E_b/N_0$ . For comparison with non permutation sequential decoding (NPSD), an equal value of

overall computational limit  $L$  was used for a same class of RS codes. The maximum number of trial,  $Y$ , in HPSD has the relation as

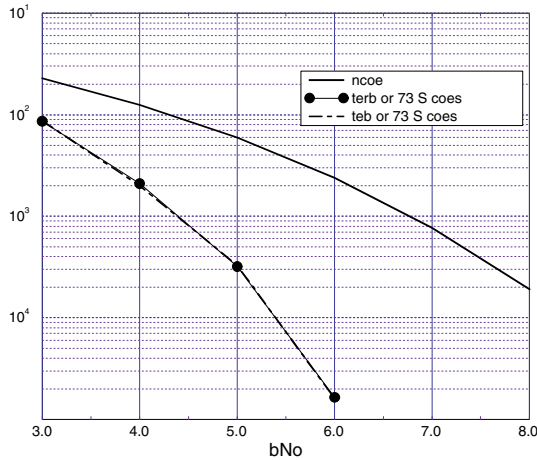
$$L = L_c \cdot Y \quad (5)$$

where  $L_c$  is a computational limit per each trial.

Fig.3 and Fig.4 are the comparison between Viterbi and *hybrid permutation sequential decoding* (HPSD) for (7,3) RS codes. In the figure, we can see that the decoding performance of HPSD was almost equal to that of Viterbi decoding. On the other hand, in Fig.4, the complexity of HPSD was much lower and it rapidly decreased as  $E_b/N_0$  increased. Thus it is well verified that the HSPD is very efficient decoding method achieving ML performance for (7,3) RS codes.

Fig.5 is the decoding performance comparison between non permutation decoding and hybrid permutation decoding. The hybrid permutation decoding produced considerable coding gain for (15,9) and (31,27) RS codes. Moreover more gain has been achieved for (31, 27) RS codes.

Fig.6 is the complexity comparison between HPSD and NPSD. In the figure, we can see that HPSD provides considerably low complexity for (7,3), (15,9) and (31,27) RS codes. In particular, when we consider the result obtained in Fig.5, the most cost-effective performance has been achieved for (31,27) RS codes. That is, HPSD provided around 1.0 dB gain with 1/3 complexity compared with NPSD. This results from the fact that more permutation group are available for (31, 27) RS codes than other two RS codes. Therefore HPSD will be more effective for long RS codes with large Galois field.



**Fig.3.** Performance Comparison with Viterbi decoder

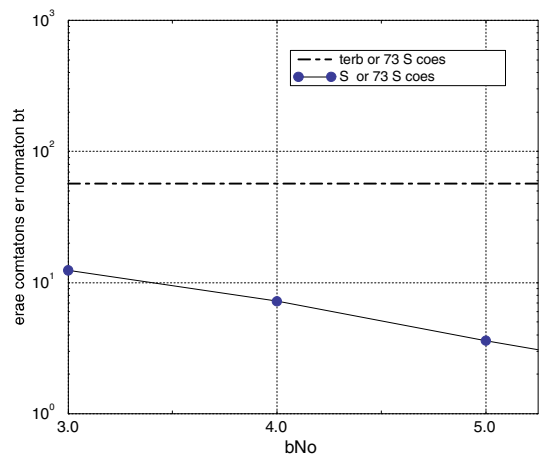


Fig.4. Complexity Comparison with Viterbi decoding

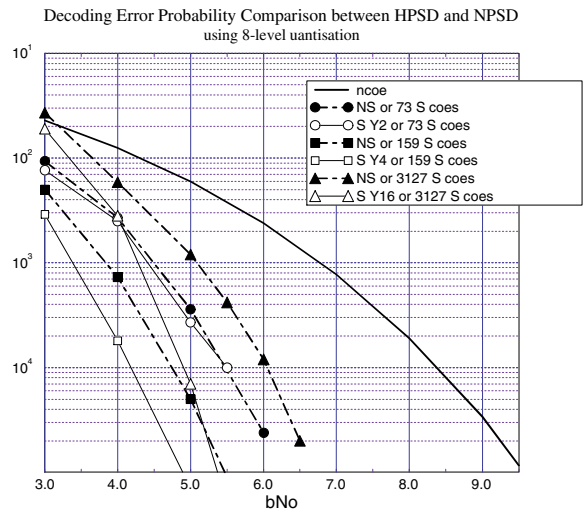


Fig.5. Decoding Performance by Permutations

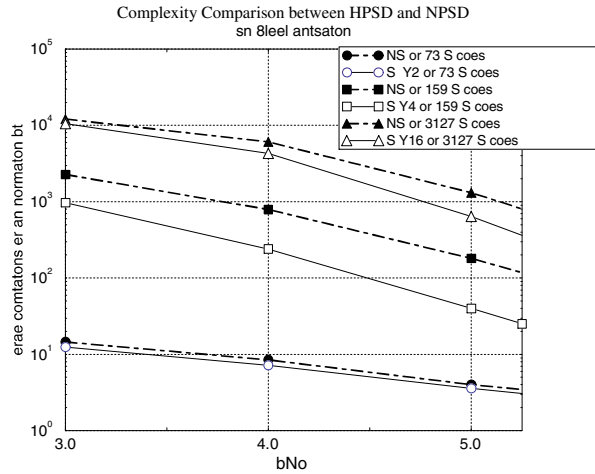


Fig.6. Complexity Comparison by Permutations

## 5 Conclusion

The use of the hybrid permutation gives a great improvement in decoding complexity and decoding performance. Since the complexity of the sequential decoding depends on the efficiency to search for the correct path at a given computational limit, the hybrid permutation decoding is very proper to drive the searching region of the decoder to the most likely one. Thus if the correct path has been tried at least once, this HPSD will always produce the maximum likelihood performance at low complexity. Furthermore this hybrid permutation can be useful to design a low complexity decoding for any block codes where the cyclic and squaring permutation are available.

## References

1. Fano, R.: A Heuristic Discussion of Probabilistic Decoding. IEEE Trans. Inform. Theory. **IT-9**, (1963) 64-74
2. Wolf, J. K.: Efficient maximum likelihood decoding of linear block codes using a trellis. IEEE Trans. Inform. Theory. **IT-20** (1978) 76-80
3. Reed, I.S. and Solomon, G.: Polynomial codes over certain finite fields," SIAM Journal on Applied Mathematics. **8** (1960) 300-304
4. Forney, G.D.: Generalized minimum distance decoding. IEEE Trans. Inform. Theory, **IT-12** (1966) 125-131
5. Chase, D.: A class of algorithm for decoding block codes with channel measurement information. IEEE Trans. Inform. Theory. **IT-18** (1972) 170-182,

6. Shin, S.: Trellis decoding of Reed-Solomon codes. Ph.D. Thesis, (1994)
7. Matis, K.R. and Modestino, J.W.: Reduced-search soft-decision trellis decoding of linear block codes. *IEEE Trans. Inform. Theory*. **39** (1991) 440-444
8. Vardy, A. and Be'ery, Y.: Bit level soft-decision decoding of Reed-Solomon codes. *IEEE Trans. Inform. Theory*. **IT-28** (1982) 349-355
9. Oh, M. and Sweeney, P.: Bit-level soft decision sequential decoding for RS codes. WCC'99. (1999) 111-120
10. Jacob, I. and Berlekamp, E.: A lower bound to the distribution of communication for Sequential Decoding. *EEE Trans. Inform. Theory*. **IT-13** (1967) 167 - 174
11. Savage, J.: The distribution of the sequential decoding computational time. *IEEE Trans. Inform. Theory*. **IT-12** (1966) 143- 147
12. Anderson, J.: Sequential decoding based on an error criterion. *IEEE Trans. Inform. Theory*. **40** (1994) 546 - 554
13. Martin, I., Honary, B., and Farrell, P.G.: Modified minimum weight decoding for RS codes. *ELECTRONICS LETTERS*. **31** (1995) 713-714.
14. Pei, D., Wang, C., and Omura, J.: Normal basis of finite field  $GF(2^m)$  . *IEEE Trans. Inform. Theory*. **IT-32** (1986) 285-287
15. Perlis, S.: Normal basis of cyclic fields of prime-power degree. *Duke Math. J.* **9** (1942) 507-517
16. Oh, M. and Sweeney, P.: Squaring permutation sequential decoding on normal basis for RS codes. *ELECTRONICS LETTERS*. **35** (1999) 1325-1326

# On Efficient Decoding of Alternant Codes over a Commutative Ring<sup>\*</sup>

Graham H. Norton and Ana Sălăgean

Algebraic Coding Research Group, Centre for Communications Research  
University of Bristol, U.K.

Graham.Norton@Bristol.ac.uk, Ana.Salagean@ntu.ac.uk

## 1 Introduction

Let  $R$  be a commutative ring e.g. the domain of  $p$ -adic integers or a Galois ring. We define alternant codes over  $R$ , which includes BCH and Reed-Solomon codes. We also define a corresponding key equation and concentrate on decoding alternant codes when  $R$  is a domain or a local ring. Our approach is based on minimal realization (MR) of a finite sequence [4,5], which is related to rational approximation and shortest linear recurrences. The resulting algorithms have quadratic complexity.

When  $R$  is a domain, the error-locator polynomial is the unique monic minimal polynomial of the finite syndrome sequence (Theorem 2), and can be easily obtained using Algorithm MR of [4] (which is division-free). The error locations and magnitudes can then be computed as over a field. In this way we can efficiently decode any alternant code over a domain.

Recall that a Hensel ring is a local ring which admits Hensel lifting. (It is well-known that a finite local ring, such as a Galois ring, is a Hensel ring.) We characterize the set of monic minimal polynomials of a finite syndrome sequence over a Hensel ring (Theorem 3). It turns out that the monic minimal polynomials coincide modulo the maximal ideal of  $R$  (Theorem 4) when  $R$  is a local ring. This yields an efficient new decoding algorithm (Algorithm 1) for alternant codes over a local ring  $R$ , once a monic minimal polynomial of the syndrome sequence is known. For determining the error locations, it is enough to find the roots of the image of any such monic minimal polynomial in the residue field  $R/\mathfrak{m}$ . After determining the error locations, the error magnitudes can be easily computed.

When  $R$  is a finite chain ring (e.g. a Galois ring) we invoke Algorithm MP of [5] to find a monic minimal polynomial.

We note that a modification of the Berlekamp-Massey algorithm for  $\mathbb{Z}_m$  was given in [8], where it was claimed [*loc. cit.*, Introduction] (without proof) to decode BCH codes defined over the integers modulo  $m$ . An algorithm to decode BCH and Reed-Solomon codes over a Galois ring has also been given in [3]. However this algorithm may require some searching see [*loc. cit.*, Conclusions,

---

<sup>\*</sup> Research supported in part by the U.K. Engineering and Physical Sciences Research Council under Grant L07680. The second author is now with Department of Mathematics, Nottingham Trent University, UK.

p. 1019] and their decoding algorithm requires root-finding in itself, which is also less efficient.

For more details and proofs, we refer the reader to [7].

## 2 Alternant Codes over a Commutative Ring

Let  $R$  be a commutative ring with  $1 \neq 0$  and let  $R^\times$  denote the subset of  $R$  consisting of all elements which are *not* zero-divisors.

The following definition of alternant codes over  $R$  generalises the definition over fields.

**Definition 1 (Alternant codes)** Let  $R$  be a subring of  $\mathbb{C}$  and  $d \geq 2$ . Suppose that  $\alpha = (\alpha_1, \dots, \alpha_n)$  and  $\beta = (\beta_1, \dots, \beta_n)$  are such that  $\alpha_i \in R^\times$  and  $\beta_i \in R^\times$  for  $1 \leq i \leq n$ . If

$$H = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1 \alpha_2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \alpha_1^2 \alpha_2 & \alpha_2^3 & \dots & \alpha_n^3 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{d-2} \alpha_2 & \alpha_2^{d-1} & \dots & \alpha_n^{d-1} \end{bmatrix} \quad (1)$$

then the alternant code of length  $n$  and alphabet  $R$  defined by  $H$  is the  $R$ -module

$$\mathcal{A}(H) = \{c \in R^n : Hc^{\text{tr}} = 0\}.$$

As usual,  $H$  is called the parity check matrix of  $\mathcal{A}(H)$ .

As in the case of fields, we have:

**Theorem 1** The minimum Hamming distance of  $\mathcal{A}(H)$  is at least  $d$ .

## 3 A Key Equation

For decoding alternant codes over a ring we follow the main steps for their algebraic decoding over a finite field, except that we rely on minimal realization of a finite sequence which was introduced in [4]. For some advantages of the minimal realization approach, see [5, Introduction]. See also the expository account in [6], especially *loc. cit.* Section 8, which discusses the application to a finite sequence of syndromes over a finite field.

Suppose that a codeword  $c \in \mathcal{A}(H)$  is received as  $\tilde{c} = c + e$ . We have to find the error vector  $e$  given the syndrome vector  $\tilde{c}^{\text{tr}} = c^{\text{tr}} + e^{\text{tr}}$ .

We will henceforth assume that  $d = 2t + 1 \geq 3$  and that the number of errors is  $t \leq \frac{d-1}{2}$ . Let  $i_1, \dots, i_w$  be the positions of the errors. As usual,  $i_1, \dots, i_w$  are called the error locations and  $e_{i_1}, \dots, e_{i_w}$  the error magnitudes. We write  $e_i$  for  $i = 1, \dots, w$ ; note that  $e_i \in R$ .

**Definition 2 (Syndrome sequence)** *The syndrome sequence of the error  $\Gamma$  is the finite sequence  $\Gamma_0 \Gamma_{-1} \Gamma \Gamma \Gamma \Gamma_m$  over  $\Gamma$ , denoted  $\Gamma\mathcal{F}$  and defined by:*

$$\Gamma_i = \sum_{k=1}^n \Gamma_k \Gamma_k \Gamma_k^{-i} = \sum_{j=1}^w \Gamma_{i_j} \Gamma_{i_j} \Gamma_{i_j}^{-i} \Gamma$$

for  $\Gamma = 0\Gamma^{-1}\Gamma\Gamma\Gamma\Gamma$ .

**Definition 3 (Error polynomials)** *We define the error-locator and error-evaluator polynomials by*

$$\Gamma_e = \prod_{j=1}^w (\Gamma - \Gamma_{i_j}) \text{ and } \Gamma_e = \sum_{j=1}^w \Gamma_{i_j} \Gamma_{i_j} \prod_{\substack{k=1, \dots, w \\ k \neq j}} (\Gamma - \Gamma_{i_k}) \Gamma$$

Note that in the classical literature  $\Gamma_e^*$  and  $\Gamma^{\deg(\sigma_e)-1-\deg(\omega_e)} \Gamma_e^*$  are called the error-locator and the error-evaluator polynomial respectively, where  $\Gamma^*$  denotes the reciprocal of  $\Gamma - \Gamma[\Gamma]$ .

**Definition 4 (Key equation)** *Let  $\Gamma = \sum_{i=m}^0 \Gamma_i \Gamma^i - \Gamma[\Gamma^{-1}]$ . We say that  $(\Gamma\Gamma) - \Gamma[\Gamma] - \Gamma \Gamma[\Gamma]$  is a solution of the key equation if  $\Gamma$  is monic,  $\deg(\Gamma) - \deg(\Gamma) - \Gamma$  and*

$$\Gamma - \Gamma\Gamma \bmod \Gamma^{m-1} \Gamma \quad (2)$$

*A solution  $(\Gamma\Gamma)$  is called minimal if  $\deg(\Gamma)$  is minimal.*

As in the classical case we easily obtain:

**Proposition 1** *If  $\Gamma - \Gamma$  then  $(\Gamma_e \Gamma \Gamma_e)$  is a solution of the key equation.*

The minimality of the solution  $(\Gamma_e \Gamma \Gamma_e)$  is not obvious, but will follow from Theorem 2 when  $\Gamma$  is a domain and from Theorem 4 when  $\Gamma$  is local.

We now recall some definitions from [5]. For  $\Gamma - \Gamma[\Gamma]$  and  $\Gamma - \Gamma[\Gamma^{-1}]$ ,  $\Gamma - \Gamma$  denotes their product in  $\Gamma[\Gamma^{-1} \Gamma]$  and  $(\Gamma - \Gamma)_j$  is the coefficient of  $\Gamma^j$  in  $\Gamma - \Gamma$ . We write  $\text{lc}(\Gamma)$  for the leading coefficient of  $\Gamma - \Gamma[\Gamma] - 0 -$ .

**Definition 5 ([5])** *Let  $\Gamma - \Gamma - 0 -$ . The  $\Gamma$ -annihilator set of  $\Gamma\mathcal{F}$  is*

$$\text{Ann}(\Gamma\mathcal{F} \Gamma) = \{-\Gamma : \text{lc}(\Gamma) = \Gamma(\Gamma - \Gamma)_j = 0 \text{ for } \Gamma + \deg(\Gamma) - \Gamma - 0 - \Gamma\}$$

A polynomial  $\Gamma$  is said to be an *annihilating polynomial of the sequence  $\Gamma\mathcal{F}$*  if  $\Gamma - \text{Ann}(\Gamma\mathcal{F} \Gamma)$  for some  $\Gamma$ .

A non-zero polynomial in  $\text{Ann}(\Gamma\mathcal{F} \Gamma)$  of minimal degree is called a *minimal polynomial of the sequence  $\Gamma\mathcal{F}$* , and we write  $\text{Min}(\Gamma\mathcal{F} \Gamma)$  for those minimal polynomials of  $\Gamma\mathcal{F}$  with leading coefficient  $\Gamma$ . (For the equivalence between minimal polynomials and shortest linear recurrences of a finite sequence, see [6, Corollary 2.3], which is valid for any  $\Gamma$ .)

Recall from [4] that for  $\Gamma - \Gamma[\Gamma]$ ,  $\Gamma(\Gamma\Gamma\Gamma\Gamma) - \Gamma\Gamma[\Gamma]$  is defined by

$$\Gamma(\Gamma\Gamma\Gamma\Gamma) = \sum_{j=1}^{\deg(f)} (\Gamma - \Gamma)_j \Gamma^j \Gamma$$

The connection between the key equation and minimal polynomials of  $\Gamma\Gamma$  becomes clear from the following lemma:

**Lemma 1** *The pair  $(\Gamma\Gamma - \Gamma[\Gamma] - \Gamma\Gamma[\Gamma])$  is a minimal solution of the key equation (2) if and only if  $\deg(\Gamma) - -\Gamma\Gamma\Gamma - \text{Min}(\Gamma\Gamma\Gamma\Gamma)$  and  $\Gamma = \Gamma(\Gamma\Gamma\Gamma\Gamma)$ .*

## 4 Decoding over a Domain

**Theorem 2** *If  $\Gamma$  is a domain then for all  $\Gamma - \Gamma - -0 -$ ,  $\text{Min}(\Gamma\Gamma\Gamma\Gamma) = -\Gamma_e -$ .*

We can now develop a decoding algorithm for alternant codes over a domain. Algorithm MR of [4] computes a minimal polynomial  $\Gamma$  and the corresponding  $\Gamma(\Gamma\Gamma\Gamma\Gamma)$  for any sequence  $\Gamma\Gamma$  over a domain. But from Theorem 2, we know that for a syndrome sequence, such a polynomial  $\Gamma$  must be the error locator polynomial multiplied by some non-zero constant. Hence, after applying Algorithm MR to the sequence of syndromes, we simply divide the output polynomials  $\Gamma$  and  $\Gamma(\Gamma\Gamma\Gamma\Gamma)$  by the leading coefficient of  $\Gamma$ , thus obtaining  $\Gamma_e$  and  $\Gamma\Gamma_e$ . The algorithm has quadratic complexity. We then proceed as in the classical (field) case: we compute the error locations as the roots of  $\Gamma_e$  (which are of the form  $\Gamma_{i_1} \Gamma \Gamma \Gamma \Gamma_{i_w}$ ) and the error magnitudes as  $\Gamma_{i_j} = \Gamma_e(\Gamma_{i_j}) \Gamma(\Gamma_e(\Gamma_{i_j}) \Gamma_{i_j})$ .

This algorithm can decode, in particular, BCH and Reed-Solomon codes over the  $\Gamma$ -adic integers of [1].

## 5 Decoding over a Local Ring

We now assume that  $\Gamma$  is a local ring with maximal ideal  $\Gamma$  and residue field  $\Gamma = \Gamma/\Gamma$ . We extend the canonical projection  $\Gamma \rightarrow \Gamma$  to a projection  $\Gamma[\Gamma] \rightarrow \Gamma[\Gamma]$  and denote the image of  $\Gamma - \Gamma[\Gamma]$  under this projection by  $\bar{\Gamma}$ .

When  $\Gamma$  is a Hensel ring we can characterize the monic minimal polynomials of the syndrome sequence:

**Theorem 3** *If  $\Gamma$  is a Hensel ring and  $\bar{\Gamma}_1 \Gamma \Gamma \Gamma \bar{\Gamma}_n$  are distinct then*

$$\text{Min}(\Gamma\Gamma\Gamma\Gamma) = \left\{ \prod_{j=1}^w ( - i_j - j ) : j i_j i_j = 0 \text{ for some } j = 1 \right\}$$

Our decoding algorithm is based on the following result:

**Theorem 4** *If  $R$  is a local ring and  $\alpha_1, \dots, \alpha_n$  are distinct then  $e = \text{Min}(\alpha_i - 1)$  and for any  $\beta = \text{Min}(\alpha_i - 1)$  we have*

$$\beta = \alpha_e = \prod_{j=1}^w (\alpha - \alpha_{i_j})$$

We can now develop a decoding algorithm for alternant codes over a local ring, provided we have an algorithm that computes a monic minimal polynomial for  $\alpha$ . The latter can be achieved for sequences of syndromes of BCH and Reed-Solomon codes over  $\mathbb{Z}_{p^a}$  (see [3], [8]), over finite local commutative rings (see [2]) and for any sequence over a finite chain ring (see [5]). A method of computing the error once we have a monic minimal polynomial  $\beta$  is discussed in [2,3]: (i) the roots of  $\beta$  in  $R$  are found and (ii) the ones that differ from some  $\alpha_i$  by a zero-divisor are selected. Our method searches for the roots of  $\beta$  [2] among  $\alpha_1, \dots, \alpha_n$  and is therefore more efficient.

**Algorithm 1 (Decoding  $\mathcal{A}(\alpha, y, d)$  over a local ring)**

*Input:*  $y = (y_1, \dots, y_n)$  containing at most  $t = (n - 1)/2$  errors, where  $\alpha_i \neq 0$ .  
*Output:*  $e = (e_1, \dots, e_n)$ , the nearest codeword.

0. Let  $w = 1 - 2t$ .
1. Compute the syndrome sequence  $s = (s_0, \dots, s_{n-1})^{\text{tr}} = y \alpha^{\text{tr}}$ . If  $s = (0, \dots, 0)$ , return  $e = (0, \dots, 0)$ .
2. Compute a monic minimal polynomial  $\beta$  for the sequence  $s$ .
3. Compute the roots  $\alpha_{i_1}, \dots, \alpha_{i_w}$  of  $\beta$  in  $R$ . Then the errors occurred at positions  $i_1, \dots, i_w$ .
4. Compute  $\alpha_e = \prod_{j=1}^w (\alpha - \alpha_{i_j})$ .
5. Compute  $e'_e$  and  $e_e = (e_e, \dots, e_n)$ .
6. Set  $e = (0, \dots, 0)$  and for  $i = 1, \dots, n$ , put  $e_i = e_e(\alpha_{i_j}) - (e'_e(\alpha_{i_j}) - \alpha_{i_j})$ . Return  $e$ .

Algorithm 1 can decode, in particular, BCH and Reed-Solomon codes over Galois rings.

*Acknowledgement.* The authors gratefully acknowledge financial support from the U.K. Engineering and Physical Sciences Research Council (EPSRC). The second author was supported by EPSRC Grant L07680.

## References

1. A. R. Calderbank and N. J. A. Sloane. Modular and  $p$ -adic codes. *Designs, Codes and Cryptography*, 6:21–35, 1995.
2. A. A. de Andrade and R. Palazzo, Jr. Construction and decoding of BCH codes over finite commutative rings. *Linear Algebra and its Applications*, 286:69–85, 1999.

3. J. C. Interlando, R. Palazzo, and M. Elia. On the decoding of Reed-Solomon and BCH codes over integer residue rings. *IEEE Trans. Inform. Theory*, 43(3):1013–1021, 1997.
4. G. H. Norton. On the minimal realizations of a finite sequence. *J. Symbolic Computation*, 20:93–115, 1995.
5. G. H. Norton. On minimal realization over a finite chain ring. *Designs, Codes and Cryptography*, 16:161–178, 1999.
6. G. H. Norton. On shortest linear recurrences. *J. Symbolic Computation*, 27:323–347, 1999.
7. G. H. Norton and A. Sălăgean. On the key equation over a commutative ring. *Designs, Codes and Cryptography*, 1999. To appear.
8. J. A. Reeds and N. J. A. Sloane. Shift-register synthesis (modulo  $m$ ). *SIAM J. Computing*, 14:505–513, 1985.

# Reduced Complexity Sliding Window BCJR Decoding Algorithms for Turbo Codes

Jihye Gwak<sup>1</sup>, Sooyoung Kim Shin<sup>2</sup>, Hyung-Myung Kim<sup>3</sup>

<sup>1</sup>Satellite Communications System Department, Electronics and Telecommunications Research Institute, 161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Korea  
[jihye@etri.re.kr](mailto:jihye@etri.re.kr)

<sup>2</sup>Satellite Communications System Department, Radio & Broadcasting Technology Laboratory, ETRI, 161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Korea  
[dssy@satnet.etri.re.kr](mailto:dssy@satnet.etri.re.kr)

<sup>3</sup>Department of Electrical Engineering, Korea Advanced Institute of Science and Techonology, 373-1 Kusong-Dong, Yusong-Gu, Taejon, 305-701, Korea  
[hmkim@panda.kaist.ac.kr](mailto:hmkim@panda.kaist.ac.kr)

**Abstract.** In decoding the turbo codes, the sliding window BCJR algorithm, derived from the BCJR algorithm, permits a continuous decoding of the coded sequence without requiring trellis termination of the constituent codes and uses reduced memory span. However, the number of computations required is greater than that of BCJR algorithm. In this paper, we propose an efficient sliding window type scheme which maintains the advantages of the conventional sliding window algorithm, reduces its computational burdens, and improves its BER performance by allowing the window to be forwarded in multi-step. Simulation results show that the proposed scheme outperforms the conventional sliding window BCJR algorithm with reduced complexity.

## 1 Introduction

The decoding of turbo codes is performed frame by frame assuming that the receiver knows the final states of each frame [1], [2]. It means that the turbo encoder requires trellis termination. However, the trellis termination of turbo codes is non-trivial unlike convolutional codes [3].

The sliding window (SW) BCJR algorithm for continuous decoding is proposed by Benedetto *et al.* which does not divide information bits into blocks and does not require trellis termination [2]. The SW BCJR algorithm has an advantage in the application where it requires a small delay such as speech transmission with short frames, since trellis termination usually requires another redundancy. However, the computational complexity of SW BCJR algorithm is even greater than that of the BCJR algorithm which also suffers from high computational burdens. Therefore it is essential to reduce the computational complexity of the SW BCJR algorithm.

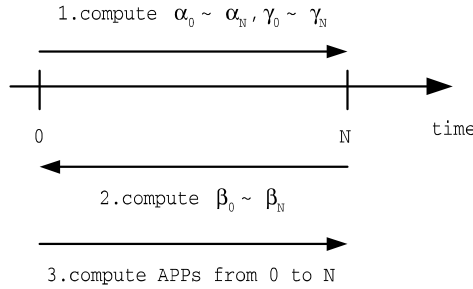
In this paper, we propose an efficient sliding window type scheme which reduces the complexity by forwarding the window by  $C$  steps, where  $C \geq 1$ . The proposed algorithm resulted in enhanced performance compared to the SW BCJR algorithm with the same complexity.

In section II we describe SW BCJR algorithm compared to BCJR algorithm, and in section III we explain an efficient sliding window type algorithm to overcome the complexity problem of conventional SW BCJR algorithm. Section IV is dedicated to simulation results for different decoding algorithms. Finally conclusion is drawn in section V.

## 2 BCJR Algorithm and SW BCJR Algorithm

The BCJR algorithm estimates *a posteriori* probability (APP) of information bit to obtain log likelihood ratio (LLR) [1]. In this paper, we do not detaily describe the numerical expressions for LLRs, and we simply adopt the expressions used in [2]. LLRs typically represent soft outputs, which are used in an iterative decoding process. The BCJR algorithm calculates  $\alpha_k(S_i), \beta_k(S_i)$  by forward and backward recursion respectively [2], and  $\gamma_k(c)$  whenever the channel outputs of codewords are received [2], where  $k$  is a time index,  $S_i$  represents encoder state, and  $c$  is a codeword which is determined by encoder state and information bit. Then APPs can be obtained from  $\alpha_k(S_i), \beta_k(S_i), \gamma_k(c)$ .

The BCJR algorithm requires the whole sequence to be received before the decoding process is started, and trellis terminations of each frame is also required in prior to initialize the backward recursion as shown in Fig. 1. Moreover, it is necessary for the BCJR algorithm to store all the values of  $\alpha_k(S_i)$  and  $\gamma_k(c)$  in one frame.

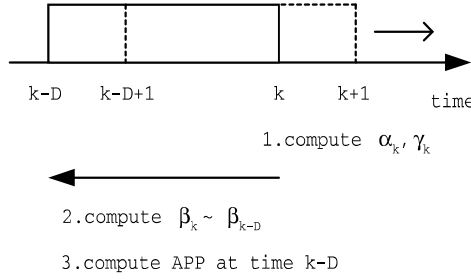


**Fig. 1.** The steps of the BCJR algorithm

In 1996, the SW BCJR algorithm is proposed by Benedetto *et al.* [2] which avoids the problem of trellis termination and operates on a fixed memory span. Forwarding the window of width  $D$ , the SW BCJR algorithm gives LLRs, but does not divide the received sequence into blocks, as shown in Fig. 2.

The SW BCJR algorithm initializes the backward recursion at time  $k$  using the value of  $\alpha_k(S_i)$ , and performs backward recursion from time  $k-1$  back to time  $k-D$  and then computes APP at time  $k-D$ . After calculating APP, the SW BCJR algorithm forwards the window by 1 step and repeats the same operations at time  $k+1$  to obtain APP at time  $k-D+1$ . Therefore the decoding process is not performed by frame basis, and also the trellis termination is not necessary. Moreover, the SW BCJR algorithm uses less

memory because it stores  $\alpha_k(S_i)$ s and  $\gamma_k(c)$ s for a corresponding window instead of a whole frame.



**Fig. 2.** The steps of the SW BCJR algorithm

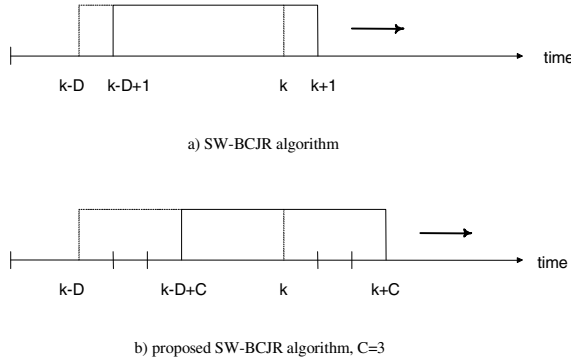
### 3 Reduced Complexity SW BCJR Algorithm

The numbers of computations of  $\alpha_k(S_i)$  and  $\gamma_k(c)$  are equal both in the SW BCJR and in the ordinary BCJR algorithms, but the required computations of  $\beta_k(S_i)$  of the SW BCJR algorithm are  $D$  times as many as those of the BCJR algorithm.

The SW BCJR algorithm performs backward recursion from  $k$  to  $k-D$  to obtain  $\beta_{k-D}(S_i)$ , and then the window of width  $D$  forwards by 1 step, as shown in Fig. 3-a). In this paper, we propose an efficient sliding window type scheme which reduces the number of computations of  $\beta_k(S_i)$  by forwarding the window by  $C$  steps (Fig. 3-b)).

If the proposed algorithm uses the window of the same width as that of the SW BCJR algorithm and  $C$  is greater than 1, the complexity of the proposed algorithm reduces but the performance degrades. This is because the initialization of backward recursion is less accurate than in the original SW BCJR. That is, the more backward recursion, the more accurate value of  $\beta_k(S_i)$  could be achieved. With the same computational complexity, however, the performance of the proposed algorithm can be enhanced. It should be noted that we can lengthen the window width of the proposed algorithm compared to that of the original SW BCJR, resulting the same computational complexity.

Let us compare the complexities of the BCJR algorithm, SW BCJR algorithm, and the proposed algorithm. These algorithms have the same numbers of computations for  $\alpha_k(S_i)$  and  $\gamma_k(c)$ , but the different numbers of computations for  $\beta_k(S_i)$ . The required computations of  $\beta_k(S_i)$  of the SW BCJR algorithm are  $D$  times as many as those of the BCJR algorithm, and the number of computations of the proposed algorithm is  $D/C$  times as those of the BCJR algorithm as shown in Table 1, where  $k_0$  represents parameter of  $(k_0, n_0)$  convolutional codes, and  $N_s$  is the number of states. In addition, the SW BCJR algorithm and the proposed algorithm have the same memory requirements, which are  $D/N$  times as many as those of the BCJR algorithm. Table 2 shows the memory requirement of each algorithms.



**Fig. 3.** The movement of window

**Table 1.** The number of computation to obtain  $\beta_k(S_i)$

	the number of additions of $2^{k_0}$ numbers each	the number of multiplications
BCJR	$N_s$	$N_s \times 2^{k_0}$
SW BCJR	$D \times N_s$	$D \times N_s \times 2^{k_0}$
proposed	$\frac{D}{C} \times N_s$	$\frac{D}{C} \times N_s \times 2^{k_0}$

**Table 2.** The memory requirement of each algorithms

	the number of $\gamma_k(c)$ to be stored	the number of $\alpha_k(S_i)$ to be stored
BCJR	$N \times M$	$N \times N_s$
SW BCJR	$D \times M$	$D \times N_s$
proposed	$D \times M$	$D \times N_s$

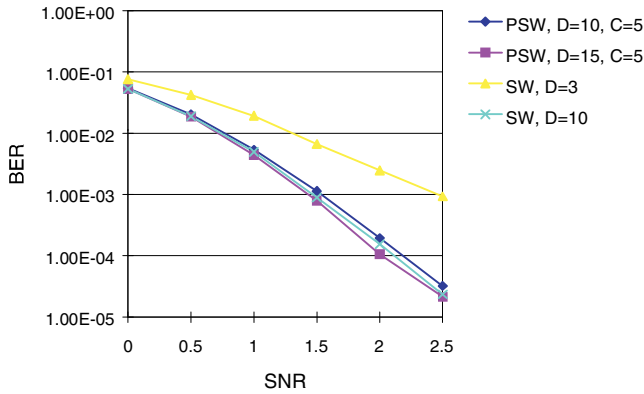
## 4 Simulation Results

In this section, we have estimated the performances of the proposed algorithm in comparison to the SW BCJR algorithm. We carried out Monte Carlo simulations using rate 1/3 turbo encoders over AWGN channel. Two types of equal component codes were employed, generator polynomials  $\{7,5\}_8$  with  $K=3$  (code 1), and  $\{17, 15\}_8$  with  $K=4$  (code 2). The simulation results are shown in Fig. 4 and Fig. 5.

In the figures, the ‘SW’ denotes sliding window BCJR algorithm, ‘PSW’ denotes proposed algorithm, and  $D$ ,  $C$  represent window width and window forwarding step

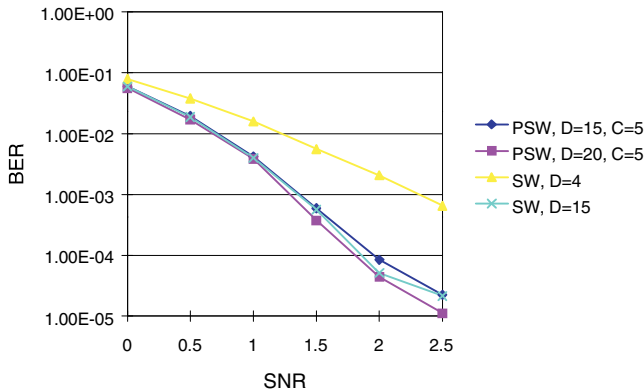
size respectively. The number of iterations in the decoding process is 2. The width of window used referred to the decoding depth of convolutional codes, which is about 5 times of the number of registers in encoder [4], [5].

Fig. 4 shows the performance comparison of the proposed algorithm for  $D=10$ ,  $C=5$  and  $D=15$ ,  $C=5$  to those of the SW BCJR algorithm for  $D=3$ , 10 with  $K=3$ . The SW BCJR algorithm with  $D=3$  and the proposed algorithm with  $D=15$ ,  $C=5$  have same complexity. The proposed algorithm with  $D=15$ ,  $C=5$  shows the best performance.



**Fig. 4.** BER performance for various decoders of code 1

Fig. 5 compares the performances of the proposed algorithm with those of the SW BCJR algorithm for  $K=4$ .



**Fig. 5.** BER performance for various decoders of code 2

## 5 Conclusions

In this paper, we propose an efficient sliding window type scheme which maintains the advantages of the conventional sliding window algorithm, reduces its computational burdens. We can improve performance and reduce complexity simultaneously with proper choices of  $C$  and  $D$ .

## References

1. C. Berrou, A. Glavieux, and P. Thitimajshma, „Near Shannon limit error-correction coding and decoding : Turbo-codes,“ in *Proc. ICC*, pp. 1064-1070, Geneva, Switzerland, May 1993.
2. S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, „Soft-output decoding algorithms for continuous decoding of parallel concatenated convolutional codes,“ in *Proc. ICC*, pp. 112-117, Dallas, U. S. A., June 1996.
3. P. Robertson, „Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes,“ in *Proc. GLOBECOM*, pp. 1298-1303, San Francisco, U. S. A., Nov. 1994.
4. F. Hemmati and D. J. Costello, Jr., „Truncation error probability in Viterbi decoding,“ *IEEE Trans. Commun.*, vol. 25, pp. 530-532, May 1977.
5. S. Lin and D. J. Costello, Jr., *Error Control Coding*. Prentice-Hall, 1983.

# Advanced Encryption Standard (AES) - An Update [Invited Paper]

Lars R. Knudsen

University of Bergen, Norway

**Abstract.** On January 2, 1997, the National Institute of Standards and Technology in the US announced that they intend to initiate the development of a new world-wide encryption standard to replace the Data Encryption Standard (DES). A call for candidates was announced world-wide with the deadline of 15th June 1998. Totally, 15 candidates were submitted from the US, Canada, Europe, Asia and Australia. The author is the designer of one of the candidates, and a codesigner of another proposal.

The AES proposals are required to support at least a block size of 128 bits, and three key sizes of 128, 192, and 256 bits. The hope of NIST is that the end result is a block cipher “with a strength equal to or better than that of Triple-DES and significantly improved efficiency.”

In March 1999 the first AES workshop was held in Rome, Italy. August 9, 1999, NIST announced the selection of five candidates for a final round of analysis. After a second AES workshop to be held in New York in April 2000, NIST intends to make a final selection of one or two algorithms for the Advanced Encryption Standard during the summer of year 2000.

The five algorithms selected to the final round are MARS, RC6, Rijndael, Serpent, and Twofish, which also are the candidates predicted by the author in a letter to NIST.

The winner(s) of the AES competition are likely to be used widely and for many years to come. Therefore, it is important that a candidate is chosen with a high level of security not only now, but also in 25 years time or more. It is of course impossible to predict which of the five candidates will survive attacks for such a long period, but this also speaks in favor of the choice of a candidate with a large security margin.

All AES candidates are iterated ciphers, where a ciphertext is computed as a function of the plaintext (and possibly some previous ciphertexts) and the key in a number of rounds. In the call for candidates NIST did not allow for a variable number of rounds. Although NIST allowed for possible “tweaks” (small changes), at the end of the first round (April 15, 1999) none of the designers changed the number of rounds of their algorithms. In fact of the five final ones, only the MARS designers suggested a modification to overcome a small key-schedule problem.

In our opinion, the number of rounds fixed by some of the designers is too small, and the algorithms will prove inadequate for long-term security. We believe that this narrows down the five candidates to only a few.

# The Piling-Up Lemma and Dependent Random Variables

Zsolt Kukorelly

kukorell@isi.ee.ethz.ch

**Abstract.**

*modulo*

## 1 The Piling-Up Lemma

In a linear cryptanalysis attack on iterated block ciphers, one identity important for the computation of the probability of success of the attack is Matsui's Piling-up Lemma, which states that for independent, binary-valued random variables  $X_1, \dots, X_n$ , the probability that  $X_1 \oplus \dots \oplus X_n = 0$  is  $1/2 + 2^{n-1} \prod_{i=1}^n (\mathbb{P}[X_i = 0] - 1/2)$  [4]. Using the notation introduced by Harpes, Kramer and Massey [2], this can be written as  $(X_1 \oplus \dots \oplus X_n) = \prod_{i=1}^n (X_i)$ , where  $(X_i) = |2 \mathbb{P}[X_i = 0] - 1|$  is the *imbalance* of the binary-valued random variable  $X_i$ .

Another important figure in this attack is that of an input/output sum. An  $n$ -round input/output sum (I/O sum) is an expression of the form  $X^{1\dots i} = P_0(X) \oplus Y_i(Y)$ , where  $X$  is the plaintext,  $Y$  is the output of the  $i$ th round of the cipher, and  $P_0, Y_i$  are binary-valued balanced functions, that is, functions which take on each of the values 0 and 1 for half of their arguments.

Now one can also define imbalances based on conditional probabilities. For an  $n$ -round I/O sum  $X^{1\dots i}$ , one defines

- the *key-dependent imbalance* as the imbalance of  $X^{1\dots i}$  given fixed values of the round keys, i.e.,  $(X^{1\dots i} | K_1, \dots, K_i) := 2 \mathbb{P}[X^{1\dots i} = 0 | K_1, \dots, K_i] - 1$ , where  $K_1, \dots, K_i$  are the round keys;
- the *average-key imbalance* as the expected value, taken over all round keys, of the key-dependent imbalances, i.e.,  $\overline{(X^{1\dots i})} := \mathbb{E}[(X^{1\dots i} | K_1, \dots, K_i)]$ .

It turns out that, provided some assumptions, the probability of success of an attack using linear cryptanalysis, that is, the probability that the key found is

the right one, is approximately proportional to the square of the average-key imbalance of the  $(r-1)$ -round I/O sum used in the attack [1,2]. Thus, it is important for the cryptanalyst to find balanced functions  $f_0$  and  $f_{r-1}$  such that  $\prod_{i=1}^{r-1} f_i$  is as large as possible.

But it is usually infeasible to compute  $\prod_{i=1}^{r-1} f_i$  as it requires the computation of  $f_i$  for all values of  $x$  and all values of the round keys. An efficient way out of this dead-end can be found in [1,2]: define  $f_i = f_{i-1}(f_{i-1}) - f_i(f_i)$ , where  $f_{i-1}$  and  $f_i$  are binary-valued functions the first two of which are balanced. Then  $\prod_{i=1}^{r-1} f_i = \prod_{i=1}^{r-1} (f_{i-1}(f_{i-1}) - f_i(f_i))$  and  $\prod_{i=1}^{r-1} f_i$  is large assures that the average-key imbalance of the corresponding I/O sum is large.

However, for the same reason as for  $\prod_{i=1}^{r-1} f_i$ , it is also usually infeasible to compute  $\prod_{i=1}^{r-1} f_i$ . If  $f_1, \dots, f_{r-1}$  were independent, then, by the Piling-up Lemma,

$$\prod_{i=1}^{r-1} f_i = \prod_{i=1}^n (f_i) \quad (1)$$

The right side of the equation can be computed much more easily because  $f_i$  involves only the input, the output and the round key of a single round. The problem is therefore reduced to finding  $f_1, \dots, f_{r-1}$  independent. This is very difficult in practice. Thus, what one usually does is to assume that  $f_1, \dots, f_{r-1}$  are independent and to apply (1). We call that the *piling-up approximation*.

The ignorance whether the piling-up approximation is valid or not, i.e., whether  $\prod_{i=1}^n (f_i)$  is a fairly accurate or a very bad approximation of  $\prod_{i=1}^{r-1} f_i$ , has never prevented anyone of using linear cryptanalysis. It is important only in the computation of the probability of success and, if the approximation is valid, it gives the cryptanalyst a clear conscience.

## 2 The Piling-Up Approximation Is Dangerous

In a sense, the piling-up approximation can be strongly misleading, as shows the following lemma.

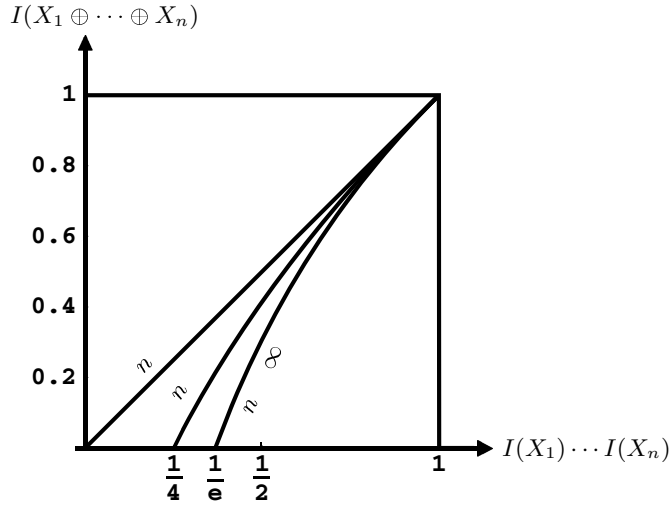
**Lemma 1.** *For any binary-valued random variables  $f_1, \dots, f_n$ , we have*

$$\prod_{i=1}^n f_i = \prod_{i=1}^n (f_i) - 2^{-n}((f_1 - 1) + (f_1 - f_n))^n \quad (2)$$

*with equality if and only if  $(f_i) = \frac{1}{n}((f_1 - 1) + (f_1 - f_n))$  for all  $i$ . Moreover, equality can occur in (2).*

*Proof.* The steps in the proof are the following (details can be found in [3]):

1.  $(f_1) + (f_2) - 1 + (f_1 - f_2)$  for all binary-valued random variables  $f_1$  and  $f_2$ ;



**Fig. 1.** For  $n \geq 2$ , all points on or above the line are possible.

- by induction,  $\binom{n-1}{k} + \binom{n-1}{n-k} = \binom{n}{k} + \binom{n}{n-k}$  for all binary-valued random variables  $X_1, \dots, X_n$ ;
- because  $\binom{n}{k} = \frac{(a_1 + \dots + a_n)^n}{n!}$  for any nonnegative numbers  $a_1, \dots, a_n$  with equality if and only if all  $a_i$  are equal, we have

$$\begin{aligned} \binom{n-1}{k} + \binom{n-1}{n-k} &= \frac{(a_1 + \dots + a_n)^{n-1}}{(n-1)!} + \frac{(a_1 + \dots + a_n)^{n-1}}{(n-1)!} \\ &= \frac{(a_1 + \dots + a_n)^{n-1}}{(n-1)!} \left( \binom{n-1}{k} + \binom{n-1}{n-k} \right) \end{aligned}$$

with equality everywhere if and only if  $(a_i) = \frac{1}{n}((n-1) + (n-1 - \dots - n))$  for all  $i$ ;

- equality can occur in (2). —

Figure 1 visualises the inequality (2). The Piling-up Lemma says that if the random variables are independent, then we are always on the diagonal. Inequality (2) says that, for  $n \geq 2$ , all points on or above the solid line are possible. Hence, for dependent random variables, the product of the imbalances can differ considerably from the imbalance of the sum and thus (1) is sometimes very far from being satisfied.

### 3 The Piling-Up Approximation Is Applicable

Fortunately for the cryptanalyst, on average, things are different.

#### 3.1 Two Random Variables

Consider two random variables  $X_1$  and  $X_2$  defined on some sample space  $\Omega$ . Let  $\Omega$  have an even number  $2$  of elements. (The case  $|\Omega|$  odd is not interesting for

our purpose.) Then  $(x_1)$  and  $(x_2)$  are of the form  $x_1 = \sum_{i=1}^n a_i x_i$  and  $x_2 = \sum_{i=1}^n b_i x_i$ , respectively, where  $a_i$  and  $b_i$  are integers,  $0 \leq a_i, b_i \leq 1$ .

Now fix  $x_1$  and  $x_2$  and consider all different pairs of random variables  $(x_1 - x_2)$ , independent or not, for which  $(x_1) = x_1$  and  $(x_2) = x_2$ . (Two random variables are different if they differ as functions from  $\Omega$  to  $[-1, 1]$ , not if their probability distribution is different.) Then compute  $(x_1 - x_2)^2$  for each pair  $(x_1 - x_2)$ . By some counting arguments, the average of  $(x_1 - x_2)^2$  is equal to  $(x_1 - x_2)^2$ , where

$$(x_1 - x_2)^2 := \frac{1}{2^{\binom{2\vartheta}{\vartheta+j}}} \sum_{m=i+j}^{\vartheta+i} \left( 2^{\binom{2\vartheta}{\vartheta+j}} - \binom{2\vartheta}{\vartheta+j} \left( \binom{2\vartheta}{\vartheta+j} - \binom{2\vartheta}{\vartheta+j-1} \right) \right)$$

and  $(x_1 - x_2)^2 := (x_1 - x_2)^2$  if  $x_1 = x_2$ .

One shows that  $(x_1 - x_2)^2 = (x_1)^2 + (x_2)^2 + (1 + (x_1 - x_2)^2) \frac{1}{2^{\vartheta-1}}$  for all  $x_1, x_2$ , where  $\lim_{\vartheta \rightarrow \infty} \frac{1}{2^{\vartheta-1}} = 0$ . This means that the average of  $(x_1 - x_2)^2$  is lower-bounded by  $(x_1)^2 + (x_2)^2$  and upper-bounded by  $(x_1)^2 + (x_2)^2 + (1 + (x_1 - x_2)^2) \frac{1}{2^{\vartheta-1}}$ . Now if the sample space  $\Omega$  on which  $x_1$  and  $x_2$  are defined is large, then the above average of  $(x_1 - x_2)^2$  is close to  $(x_1)^2 + (x_2)^2$ . This is a first indication that the piling-up approximation might be valid after all.

The same average of  $(x_1 - x_2)^2$  is equal to  $(x_1 - x_2)^2$ , where

$$(x_1 - x_2)^2 := \frac{1}{2^{\binom{2\vartheta}{\vartheta+j}}} \sum_{m=i+j}^{\vartheta+i} \left( 2^{\binom{2\vartheta}{\vartheta+j}} - \binom{2\vartheta}{\vartheta+j} \left( \binom{2\vartheta}{\vartheta+j} - \binom{2\vartheta}{\vartheta+j-1} \right) \right)$$

and  $(x_1 - x_2)^2 := (x_1 - x_2)^2$  if  $x_1 = x_2$ . One has  $(x_1 - x_2)^2 = \frac{1}{2^{\vartheta-1}} + \frac{2\vartheta}{2^{\vartheta-1}} \frac{i^2}{\vartheta^2} \frac{j^2}{\vartheta^2} - \frac{1}{2^{\vartheta-1}} - \left( \frac{i^2}{\vartheta^2} + \frac{j^2}{\vartheta^2} \right)$ , that is, the average of  $(x_1 - x_2)^2$  is  $\frac{1}{2^{\vartheta-1}} + \frac{2\vartheta}{2^{\vartheta-1}} \frac{i^2}{\vartheta^2} \frac{j^2}{\vartheta^2} - \frac{1}{2^{\vartheta-1}} - \left( \frac{i^2}{\vartheta^2} + \frac{j^2}{\vartheta^2} \right)$ . If  $\vartheta$  gets large, this is approximately equal to  $\frac{1}{2^{\vartheta-1}} - \left( \frac{i^2}{\vartheta^2} + \frac{j^2}{\vartheta^2} \right)$ . Thus, we can conclude:

**Proposition 2.** *Let  $\Omega$  be some sample space with  $2^\vartheta$  elements. Then, if  $\vartheta$  is large enough,  $(x_1 - x_2)^2 \approx (x_1)^2 + (x_2)^2$  for virtually all binary-valued random variables  $x_1$  and  $x_2$ .*

### 3.2 More than Two Random Variables

For more than two random variables, we can compute similar averages recursively. Take some integers  $0 \leq i_1, \dots, i_n \leq \vartheta$  and consider all  $n$ -tuples  $(x_1, \dots, x_n)$  of random variables such that  $(x_1) = x_1$ ,  $\dots$ ,  $(x_n) = x_n$ . Then compute  $(x_1 - \dots - x_n)^2$ . Denote by  $(x_1, \dots, x_n)$  the empirical probability that  $(x_1 - \dots - x_n) = \dots$  given that  $(x_1) = x_1, \dots, (x_n) = x_n$ ; denote also by  $(x_1, \dots, x_n)$  (resp.  $(x_1, \dots, x_n)$ ) the average of  $(x_1 - \dots - x_n)^2$  (resp. of  $(x_1 - \dots - x_n)^2$ ), that is,  $(x_1, \dots, x_n) = \binom{2\vartheta}{i_1, \dots, i_n} (x_1, \dots, x_n)$  and  $(x_1, \dots, x_n) = \binom{2\vartheta}{i_1, \dots, i_n} (x_1, \dots, x_n)$ . One shows then that

$$\begin{aligned} - \binom{1}{1} \binom{n}{n} &= \binom{k}{k} \binom{1}{1} \binom{n}{n-1}; \\ - \binom{1}{1} \binom{n}{n} &= \binom{k}{k} \binom{1}{1} \binom{n}{n-1}; \\ - \binom{1}{1} \binom{n}{n} &= \binom{k}{k} \binom{1}{1} \binom{n}{n-1}. \end{aligned}$$

By induction, one also shows that

$$\begin{aligned} - \binom{1}{1} \binom{n}{n} - \frac{1}{1} \binom{n}{n} &= \binom{1}{1} \binom{n}{n} + (-1)(1 + \binom{1}{1}) \binom{n}{n-1}, \text{ where } \binom{1}{1} \text{ is the same} \\ &\text{function as above with } \lim_{\vartheta \rightarrow 0} \binom{1}{1} = 0; \\ - \binom{1}{1} \binom{n}{n} &= \frac{1}{2\vartheta-1} + \frac{2\vartheta}{2\vartheta-1} \frac{i_n^2}{\vartheta^2} \binom{1}{1} \binom{n}{n-1} - \frac{1}{2\vartheta-1} \left( \binom{1}{1} \binom{n}{n-1} + \frac{i_n^2}{\vartheta^2} \binom{1}{1} \right). \end{aligned}$$

Again, if  $n$  is large but  $k$  fixed, then  $\binom{1}{1} \binom{n}{n} \approx \frac{1}{1} \binom{n}{n}$ , and from the recursion for  $\binom{1}{1}$  follows that  $\binom{1}{1} \binom{n}{n} \approx \frac{2}{1} \binom{n}{n}^2$ . Thus, if  $n$  is large, then the average of  $\binom{1}{1} \binom{n}{n}$  is close to  $\binom{n}{i=1} \binom{1}{1}$  and the average of  $\binom{2}{1} \binom{n}{n}$  is close to  $\binom{n}{i=1}^2 \binom{1}{1}$ . Hence, we have:

**Theorem 3.** *Let  $\Omega$  be some sample space with  $2^n$  elements. If  $n$  is large enough, then  $\binom{1}{1} \binom{n}{n} \approx \binom{n}{i=1} \binom{1}{1}$  for virtually all binary-valued random variables  $\binom{1}{1} \binom{n}{n}$  defined on  $\Omega$ .*

### 3.3 Implication for the Piling-Up Approximation

Let  $n$  be the text blocklength of the cipher and  $k$  be the length of the round keys. The plaintext  $x_i$  and the round keys  $k_i$  are usually considered to be independent random variables uniformly distributed on  $\{-0, 1\}^n$  and  $\{-0, 1\}^k$ , respectively. Because the round functions yield an invertible function when one fixes the value of the round key, the output  $y_i$  of the  $i$ th round of the cipher is also a random variable uniformly distributed on  $\{-0, 1\}^n$ . Thus,  $(y_i - 1) \binom{1}{1} \binom{n}{n}$  is a random variable with values on  $\{-0, 1\}^{2n+k}$  and  $y_i = y_{i-1} \binom{1}{1} \binom{n}{n} - k_i \binom{1}{1} \binom{n}{n}$  is a binary-valued random variable with sample space  $\Omega = \{-0, 1\}^{2n+k}$  with  $2^{2n+k-1}$  elements. In practical ciphers,  $2^{2n+k-1}$  is fairly large. Thus, by the above Theorem,  $\binom{1}{1} \binom{n}{n} \approx \binom{r-1}{i=1} \binom{1}{1}$  in virtually all cases, that is, the piling-up approximation is valid.

## References

*Cryptanalysis of Iterated Block Ciphers*

*Advances*

*in Cryptology – Eurocrypt’95*

*On The Validity of Some Hypotheses Used in Linear Cryptanalysis*

*Advances in*

*Cryptology – Eurocrypt’93*

# A Cryptographic Application of Weil Descent

i \*1 i 2

1 r r rr

s.galbraith@rhbnc.ac.uk

2 r r r r r . .  
nigel\_smart@hpl.hp.com

## Abstract.

r  
r r r r r  
r r r r . r r  
r r r r r . r  
r r r r .

## 1 Introduction

y i yp p i i y i i  
i ppi ip i i q<sup>n</sup> i  
n > i p p i i i ppi  
i k ip i i p i  
i ik y k  
i i y i i i x-  
p i i i i p p

i i i A E q<sup>n</sup>  
2 i C q i i y A  
3 i i p k E q<sup>n</sup> Jac C q  
i i p Jac C q i i x

p iz i p p p y p  
k ip i i p i  
x p ip i i  
i i p p y p i y k q<sup>n</sup>  
i ip i i p i i  
i 2 3 i i i i  
i i i i i y p i i  
i i i i -k i  
i y y yp p i  
i y

---

\* r r .

$$\begin{array}{ccccccc}
& & & r & & & r \\
& & i & & i & p & i & & i & & 3 \\
& i & & i & & & xp & i & & pp & i \\
y & i & p & x & p & & i & x & p & & i & i \\
2^m & & ip & i & & E & & 2^{4m} & & & & \\
& i & & i & & & i & & i & p & i & i & i \\
& p & & & & & i & i & i & -xp & i & i & i \\
& & & i & & i & & p & p & & i & & \\
y & & xp & & ip & i & & & & & i & & \\
& k & p & i & & xp & i & y & i & p & i & n & y \\
& k & & p & i & & p & i & y & xp & i & i & \\
i & i & p & i & & p & i & & n & & & & 
\end{array}$$

## 2 Curves, Divisor Class Groups, and Jacobians

$$\begin{array}{ccccccc}
& & & & i & y & i \\
& & 2 & i & & & \\
\text{curve } C & & k & i & p & -i & i & y & i & i \\
k & & i & & i & i & p & i & & i \\
& p & & i & i & i & i & & i & \\
& i & & i & & & pp & & i & \\
i & i & i & C/k & i & k & & -p & p & \\
& -i & & i & p & y & i & i & & k \\
& & yC & i & -i & & C & & i & i \\
& x & 3 & i & i & & k & & & \\
i & & p & C \rightarrow C & & & & & & \\
& \text{genus } g & & -i & & i & i & p & i & i \\
& i & & y & & \text{geometric genus} & & & i & \\
i & i & i & i & i & i & & i & i & \\
& \text{Jacobian variety } \text{Jac } C & & -i & & C & & g & & \\
& k & i & i & i & y & k & i & i & g & i \\
& i & i & & g & y & i & p & C^{(g)} & & \\
& & i & p & & & i & & & & 
\end{array}$$

**Proposition 1** Suppose  $C$  is a non-singular curve over a field  $k$  with a point  $P$  defined over  $k$ . The following properties hold.

1. (Canonical map from  $C$  into  $\text{Jac } C$ ) There is a canonical map  $f^P : C \rightarrow \text{Jac } C$  which takes the point  $P$  to the identity element of  $\text{Jac } C$ .
2. (Universal property) Suppose  $A$  is an abelian variety over  $k$  and suppose there is some mapping of varieties  $\phi : C \rightarrow A$  such that  $\phi(P) = 1_A$ , then there is a unique homomorphism  $\psi : \text{Jac } C \rightarrow A$  such that  $\phi = \psi \circ f^P$ .

$$\begin{array}{ccccccc}
& i & i & & p & \text{Pic}_k^0 C & & i & & p & & z & i & i \\
C & i & & k & & k & & p & i & ip & i & i & C & i & -i \\
& & k & i & & k-p & i & P & \text{Jac } C & & \text{Pic}_k^0 C & & i & & p & i
\end{array}$$

**Proposition 2** *Let  $A$  be a simple abelian variety of dimension  $d$  over a field  $k$ . Suppose we have a map  $\phi: C \rightarrow A$  from a non-singular curve  $C$  to  $A$ . Then the genus of  $C$  is at least  $d$ . Furthermore,  $g(C) = d$  if and only if  $A$  is isogenous to the Jacobian of  $C$ .*

$$V \rightarrow \mathcal{P} - W_{K/k} \otimes E \rightarrow \mathrm{Tr}_{K/k} \mathcal{P} \rightarrow \mathcal{O} -$$

where the trace is computed using the mapping from  $W_{K/k} \otimes E$  to  $E \otimes K$ .

**Definition 1** Define  $A$  by

- i) If  $E$  is not defined over  $k$ , then set  $A = W_{K/k}(E)$ . Hence  $\# A = n$ .
- ii) If  $E$  is defined over  $k$ , then set  $A = V$ , from Lemma 1. Hence  $\# A = n - 1$ .

$$\begin{array}{ccccccc}
 & xp & i & i & y A & i p & i \\
 i i & ip i & i & i i i & y & p i & i i y \\
 i i & & p i & A & & & i p i \\
 & i y A & & & & & \\
 & y & i k & & i & k & ip i \\
 i & i p & i & ip i & E & K & \\
 i & i y A & k & x & p i y i & & C \\
 k yi & A & C & k-p i & P_0 & p i & i y A \\
 y & i p p y & i & i & i & ppi & i i i \\
 \psi Jac C^- \rightarrow A & C^{-i} & & i i & C & & \\
 & p i P_1 & P_2 & i & i p & i E K & p \\
 p i & A k i & i & y & p i & y p - k & \\
 \psi & i i i D_1 & D_2 i & Pic_k^0 C & k & pp i y & \\
 - i & p i C & \psi D_i & P_i & i y & i & i \\
 p & D_2 i & p & D_1 & Pic_k^0 C & & i x \\
 & & i p & i & & i & ppy \\
 i & & & & & & \\
 & i & y & & A & & \\
 2 i & y p & k p i & A & i i & C & \\
 3 i & y & i x & & & & i i \\
 & p & & & & & 
 \end{array}$$

## 4 Pulling Back Along $\psi$

$$\begin{array}{c}
 \begin{array}{ccccccc}
 & & i & & ppi & \psi & xp\ i\ i\ y \\
 & g & k & \phi & C \rightarrow A & & C \\
 A & pp & P_0 i & k-p\ i & C & i & i\ i\ y \\
 & p & f & f^{P_0} & i & i\ y & A \\
 & y & p & i & D & D_{eff} - d\ P_0 & D_{eff} \\
 & i & i\ i & d & Q_i & p\ i & C\ k \\
 & i\ i & D_{eff}\ i & k & i & y & d - g \\
 p & & i & k & i & y & d
 \end{array}
 \end{array}$$

**Proposition 3** *The map  $\psi : Pic_k^0 C \rightarrow A/k$  is given by*

$$\psi(D_{eff} - d\ P_0) = \sum_{i=1}^d \phi(Q_i)$$

where the addition on the right hand side is addition on the abelian variety  $A$  (which can be efficiently computed via the addition law on  $E/K$ ).

*Proof.*  $i\ i\ D_{eff} - d\ P_0\ i\ Pic_k^0 C\ i\ i$   
 $Q_i - P_0\ i\ p\ f\ C\ k \rightarrow Pic_k^0 C\ p\ p\ y$   
 $f\ Q_i\ Q_i - P_0\ ppi\ \psi\ Pic_k^0 C \rightarrow A\ i\ p\ p\ y$   
 $\phi\ \psi - f\ \psi\ Q_i - P_0\ \phi\ Q_i - A\ k\ i\ \psi\ i\ p$   
 $p\ i\ i\ p\ i\ p\ Gal\ k/k$

$$\begin{array}{c}
 \begin{array}{ccccccc}
 p & i & i & i & i & i & C \\
 i & i & p & ip\ i & p & i & C \\
 i\ i & pp & i & -i & p\ i & C & A \\
 p\ i & i & p\ \psi & i\ i & i & p & \psi \\
 p\ i & P & A & i & i\ i & y\ p & i \\
 & p\ i\ i\ i & & & & & \\
 & i & d & i\ p\ i & -Q_1, \dots, Q_d - & C\ k & y\ i \\
 p\ i & i & i & Q_i & x_i, y_i & x_i & y_i \\
 p\ i & i & i\ y\ A & i & p\ \phi & i\ T_i & \phi\ Q_i \\
 i & i & i & p\ i & i & i\ d & i \\
 k & y & i & p & A & ppi & p\ i \\
 i & i & i & i & \sum_{i=1}^d T_i & & y \\
 n & i & P & i & A & i & i \\
 & i & & & & & \\
 V & d > n & xp & i & i & 2d & k \\
 d-n & x & p & d & n & i & y \\
 p\ i & i & i\ y & i\ p & p\ i & Q_i & i \\
 -i & p\ i & C & i & i\ i & D & \sum_{i=1}^d Q_i - d\ P_0 \\
 i & i & C & i & p & \psi & i \\
 i & p\ i & i\ y\ V & i & i & i & i
 \end{array}
 \end{array}$$







r r  
 i p i y i  
 ip i yp y i p i  
 i p i i y i i p p y ppi  
 ip i  $q^n$  i  $n >$  ip i  
 p i pp i i  
 k i i i  
 p i i i i p i  
 p  
 i i i i i ppi i i  
 p i i i p ip i i  
 pp xi y  $q^n$  p i i y i i  
 p i p iz  $q^g$  g i C  
 g i y p n i i p  
 i i i p i  
 i i i i y g  
 i i i xp i y y p i y i x  
 i x q g i i y iz  
 i k k n g ik  
 fli i  
 2 i i y yi i i y A  
 i i i p g C  
 p i n i x p  
 i i i i i xp i  
 i n xp xp i C i  
 i C xp i y i n p xi y  
 i k xp i i  $q^g$  i i xp i i  
 ip i p iz  $q^n$   
 p i k i xp  
 p i C A i i i i ki  
 xp i i i i i y  
 i i i p i i i  
 C  $O n^d$  y A x d xi i  
 y i i i i i  
 i i i i y A n i i  
 i i i x p i A y i  
 A i n i i y ik y A  
 i n i i ik y i  
 i p i k  
 3 i i y p i i i y  
 i n i y i p i i n i  
 i k i i i i n i  
 i pp yp i n 2 3

ANTS-1 : *Algorithmic Number Theory*

*J. Symbolic Computation* **24**

<http://cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html>

*J. AMS* **2**

**25**

*Arithmetic Geometry*

**55**

# Edit Probability Correlation Attack on the Bilateral Stop/Go Generator

Renato Menicocci<sup>1</sup> and Jovan Dj. Golić<sup>2</sup>

<sup>1</sup> Fondazione Ugo Bordoni  
Via B. Castiglione 59, 00142 Roma, Italy  
rmenic@fub.it

<sup>2</sup> School of Electrical Engineering, University of Belgrade  
Bulevar Revolucije 73, 11001 Belgrade, Yugoslavia  
golic@galeb.etf.bg.ac.yu

**Abstract.** Given an edit transformation defined by the stop/go clocking in the bilateral stop/go generator, an edit probability for two binary strings of appropriate lengths is proposed. An efficient recursive algorithm for the edit probability computation is derived. It is pointed out how this edit probability can be used to mount a correlation attack on one of two clock-controlled shift registers. By estimating the underlying false alarm probability, it is shown that the minimum output sequence length required to be known for a successful attack is linear in the length of the shift register. This is illustrated by experimental correlation attacks on relatively short shift registers.

**Key words.** Stream ciphers, mutual clock control, bilateral stop/go, edit probability, correlation attack.

## 1 Introduction

Clock-controlled shift registers are an important tool for designing keystream generators for stream cipher applications. Several keystream generators based on clock-controlled shift registers are known to produce sequences with long period, high linear complexity, and good statistical properties (e.g., see [1]). The stop-and-go clocking is particularly appreciated in practice because of its suitability in high-speed applications. At any time, a stop/go shift register is clocked once if the clock-control input bit is equal to 1 (or 0) and is not clocked at all otherwise.

The bilateral stop/go generator (BSG) is a combination of two binary LFSRs, LFSR<sub>1</sub> and LFSR<sub>2</sub>, which mutually clock-control each other (see [3],[4]). More precisely, a clock-control function derives two clock-control bits from the states of the two LFSRs. Each clock-control bit is used to stop/go clock-control one of the LFSRs. The two clock-control bits are never simultaneously equal to zero, so that at each step at least one of the two LFSRs is stepped. The output sequence is formed as the bitwise sum of the two stop/go clocked LFSR sequences.

No attacks on such a structure are reported in the open literature. The objective of this paper is to investigate whether a divide-and-conquer correlation

attack on one of the LFSRs is possible. In such a correlation attack, the cryptanalyst would try to reconstruct the initial state of the chosen LFSR from a known segment of the keystream sequence by using an appropriate edit probability as a measure of correlation.

For the stop/go clocking, a specific edit probability correlation attack on the alternating step generator is proposed in [2]. This generator consists of two stop/go clocked LFSRs and a regularly clocked clock-control LFSR. At each time, the clock-control bit defines which of the two LFSRs is clocked, and the output sequence is obtained as the bitwise sum of the two stop/go LFSR sequences. The target of this correlation attack are the initial states of the individual stop/go clocked LFSRs.

Problems to be addressed in this paper are how to define the edit probability, how to compute it efficiently, and how to estimate the known keystream sequence length required for a successful correlation attack. The fact that the first binary derivative of the BSG output sequence is bitwise correlated to the first binary derivative of the output sequence of each of the stop/go clocked LFSR<sub>1</sub> and LFSR<sub>2</sub> suggests that a divide-and-conquer attack may be possible. The edit probability is based on an edit transformation taking into account the stop/go clocking in the BSG. By introducing a suitable partial edit probability, a recursive algorithm for computing the edit probability is derived.

Accordingly, a correlation attack on LFSR<sub>1</sub> based on the edit probability is proposed. More specifically, this edit probability is defined for two binary strings of appropriate lengths: a given input string corresponding to the output sequence of LFSR<sub>1</sub> when regularly clocked and a given output string corresponding to the first binary derivative of the output sequence of the BSG. The (random) edit transformation consists of the stop/go clocking as in the BSG of the given input string  $X$  and a purely random binary string  $Y$  (corresponding to the unknown LFSR<sub>2</sub> sequence) according to  $X$  and an auxiliary purely random and independent binary clock-control string  $R$ , of the bitwise addition of the two stop/go clocked strings, and of taking the first binary derivative of the combination string. The auxiliary binary clocking string  $R$  is introduced in order to enable a recursive computation. The edit probability is then defined as the probability that a given input string is transformed into the given output string by the described random edit transformation.

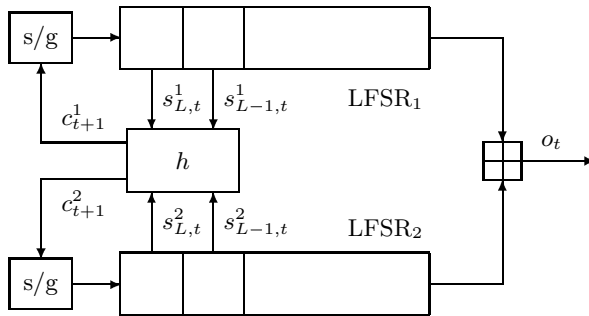
In the proposed correlation attack, for every possible LFSR<sub>1</sub> initial state, an input string of sufficient length is generated and the edit probability for a given output string is computed. The correct LFSR<sub>1</sub> initial state is then likely to belong to a set of states with the associated edit probability close to being maximal. This attack can be successful only if there is a sufficient statistical distinction between the probability distributions of the edit probability when the input string is guessed correctly and randomly, respectively. By computer simulations, for an appropriate missing event probability, the underlying false alarm probability is approximated by an exponentially decreasing function of the string length. If  $L$  denotes the common length of the two LFSRs, the minimum

output sequence length required to be known for a successful attack is then linear in  $L$ . The time complexity of the attack is then estimated as  $O(2^{L+3 \log_2 L})$ .

In Section 2, a more detailed description of the BSG is provided. The edit probability for the auxiliary clocking string and the recursive algorithm for its efficient computation are presented in Section 3. The underlying false alarm probability is estimated in Section 4 and the corresponding correlation attack is explained in Section 5. Experimental correlation attacks conducted by computer simulations are reported in Section 6. Conclusions are given in Section 7 and a number of tables displaying the statistics of the edit probability are presented in the Appendix.

## 2 Description of Bilateral Stop/Go Generator

As shown in Fig. 1, the output of the bilateral stop/go generator (BSG) is obtained by bitwise addition (modulo 2) of the output sequences of two binary linear feedback shift registers, LFSR<sub>1</sub> and LFSR<sub>2</sub>, which mutually clock-control each other by stop/go clocking (see [3],[4]). It is assumed that LFSR<sub>1</sub> and LFSR<sub>2</sub> have primitive feedback polynomials of the same degree  $L$ . At each step, the output bit is assumed to be produced in the step-then-add manner as follows. Let  $s_{L,t}^i$  and  $s_{L-1,t}^i$ ,  $i = 1, 2$ , denote the contents at step  $t - 0$  of the stages at positions  $L$  and  $L - 1$ , respectively, of LFSR <sub>$i$</sub> . From input bits  $s_{L,t}^1, s_{L-1,t}^1, s_{L,t}^2$ , and  $s_{L-1,t}^2$ , a clock-control function  $h$  determines the clock-control bits  $c_{t+1}^1$  and  $c_{t+1}^2$ . From four binary inputs  $a, b, c$ , and  $d$ ,  $h(a, b, c, d)$  outputs  $-1$ —if  $(a, b) = (0, 1)$ ,  $-2$ —if  $(c, d) = (0, 1) \equiv (a, b)$ , and  $-1, 2$ —otherwise. To get the BSG output bit  $o_t$  at step  $t$ ,  $t - 1$ , we step LFSR <sub>$i$</sub> ,  $i = 1, 2$ , or not depending on whether  $c_t^i = 1$  or  $c_t^i = 0$ , respectively, and then we add modulo 2 the output bits ( $s_{1,t}^1$  and  $s_{1,t}^2$ ) of the shift registers.



**Fig. 1.** The bilateral stop/go generator.

Some theory of the bilateral stop/go generator is exposed in [3]. In a few words, the state diagram of the BSG consists of  $3 \cdot 2^{L-2} - 1$  branched cycles each of length  $T = 5 \cdot 2^{L-2} - 1$ . At any cycle state there is at most one branch. Every

branch has length 1 and starts with a state having no predecessor. By using  $L$  such that  $T$  is prime, the linear complexity of the sequence produced while the generator covers a cycle has a lower bound of the same order of magnitude as  $T$  (see [3],[4]).

We assume that the cryptanalyst knows the feedback polynomials of the two LFSRs and operates in the known plaintext scenario. The cryptanalyst's objective is then to reconstruct the secret-key-controlled LFSR initial states from a known segment of the keystream sequence.

In the sequel, we denote by  $A$  a sequence of symbols  $a_1, a_2, \dots$  and by  $A^n$  a string  $a_1, a_2, \dots, a_n$  constituted by the first  $n$  symbols of  $A$ . When  $A$  is a binary sequence, we denote its first derivative by  $\dot{A} = \dot{a}_1, \dot{a}_2, \dots$ , where  $\dot{a}_t = a_t - a_{t+1}$ ,  $-$  standing for modulo 2 addition.

Let  $X = x_1, x_2, \dots$  and  $Y = y_1, y_2, \dots$  denote two binary input sequences and let  $C = c_1, c_2, \dots$  denote a three-valued clock-control sequence, where  $c_i \in \mathcal{C}$ ,  $\mathcal{C} = \{-1, -2, -1, 2\}$ . Let  $O = G(X, Y, C) = o_1, o_2, \dots$  denote the combination sequence produced from  $X$  and  $Y$  by the step-then-add bilateral stop/go clocking according to  $C$ , where  $X$  and  $Y$  correspond to the regularly clocked LFSR<sub>1</sub> and LFSR<sub>2</sub> sequences, respectively. Note that  $c_t$  determines which register is stepped in order to get the BSG output bit at step  $t$ .

We initially have  $o_1 = x_2 - y_1$  if  $c_1 = -1$ ,  $o_1 = x_1 - y_2$  if  $c_1 = -2$ , and  $o_1 = x_2 - y_2$  if  $c_1 = -1, 2$ . Let  $w_i(C^{s+1})$  denote the number of occurrences of the symbol  $-i$ ,  $i = 1, 2$ , in the string  $C^{s+1}$ . For simplicity, let  $w_1(C^{s+1}) = l_1$  and  $w_2(C^{s+1}) = l_2$ . The number of occurrences of the symbol  $-1, 2$  in  $C^{s+1}$  is then  $s+1-l_1-l_2$ . Thus, for any  $s \geq 0$ ,  $o_{s+1} = x_{s+2-l_2} - y_{s+2-l_1}$ . As for the generation of  $C$ , according to the BSG scheme from Fig 1, we have  $c_1 = h(x_L, x_{L-1}, y_L, y_{L-1})$  and, for  $s \geq 0$ , we can readily write  $c_{s+2} = h(x_{L+s+1-l_2}, x_{L+s-l_2}, y_{L+s+1-l_1}, y_{L+s-l_1})$ .

Consequently, for input/output strings of finite length, we have  $O^{n+1} = G^{n+1}(X^{n+2}, Y^{n+2}, C^{n+1})$ , with  $C^{n+1} = H^{n+1}(X^{n+L}, Y^{n+L})$ . Alternatively, we can write  $O^{n+1} = F^{n+1}(X^{n+L'}, Y^{n+L'})$ , where  $L' = \max(2, L)$  and  $F$  represents the joint action of  $G$  and  $H$ .

### 3 Edit Probability for Auxiliary Clocking String

In this section, we adopt a simplified model for the BSG which allows us to define and recursively compute a suitable *edit probability*. The edit probability so defined can be used for inferring about the LFSR<sub>1</sub> initial state from a given BSG output segment.

Consider an auxiliary random sequence  $R = r_1, r_2, \dots$  which is used to replace the input sequence  $Y$  in the role of generating the input bits for the clock-control function  $h$ . The simplified BSG model is as follows.  $X$ ,  $Y$ , and  $R$  are independent and purely random binary sequences (a sequence of independent uniformly distributed random variables over any finite set is called purely random). The clock-control string  $C^{n+1}$  is generated as follows. Initially, we have  $c_1 = h(x_L, x_{L-1}, r_2, r_1)$  and, for  $s \geq 0$ ,  $c_{s+2} = h(x_{L+s+1-l_2}, x_{L+s-l_2}, r_{2s+4}, r_{2s+3})$ ,

where, as above,  $l_1 = w_1(C^{s+1})$  and  $l_2 = w_2(C^{s+1})$ . We represent this by writing  $C^{n+1} = H^{n+1}(X^{n+L}, R^{2n+2})$ . The output string is generated as  $O^{n+1} = G^{n+1}(X^{n+2}, Y^{n+2}, C^{n+1})$ . The joint action of  $G$  and  $H$  is represented by  $F$  as  $O^{n+1} = F^{n+1}(X^{n+L'}, Y^{n+2}, R^{2n+2})$ , where  $L = \max(L, 2)$ .

Now we can define a suitable *random edit transformation* and an associated *edit probability*. In the given model, we start by considering the string  $\dot{O}^n = \dot{F}^n(X^{n+L'}, Y^{n+2}, R^{2n+2})$ . The transformation of a given input string  $X^{n+L'}$  into a given output string  $\dot{O}^n$ , according to a random input string  $Y^{n+2}$  and an auxiliary random clocking string  $R^{2n+2}$ , defines a random edit transformation.

Let  $Z^n = z_1, z_2, \dots, z_n$  denote a given output string. The associated edit probability for a given input string  $X^{n+L'}$  and a given output string  $Z^n$  is the probability that  $X^{n+L'}$  is transformed into  $Z^n$  by a random edit transformation according to random  $Y^{n+2}$  and  $R^{2n+2}$ . Formally, we have

$$P(X^{n+L'}; Z^n) = \Pr\{\dot{F}^n(X^{n+L'}, Y^{n+2}, R^{2n+2}) = Z^n | X^{n+L'}\}. \quad (1)$$

The statistically optimal edit probability (minimizing the error probability when deciding on  $X^{n+L'}$  given  $Z^n$ ) is then given as

$$\Pr\{X^{n+L'} | \dot{F}^n(X^{n+L'}, Y^{n+2}, R^{2n+2}) = Z^n\} = P(X^{n+L'}; Z^n) \Pr\{X^{n+L'}\}. \quad (2)$$

As  $\Pr\{X^{n+L'}\} = 2^{-(n+L')}$ , the edit probability (1) is also statistically optimal.

Our objective is to examine whether the defined edit probability can be computed efficiently by a recursive algorithm whose computational complexity is significantly smaller than  $O(2^{3n+4})$ , which corresponds to the computation of (1) by the summation of the elementary probability  $2^{-(3n+4)}$  over all  $Y^{n+2}$  and  $R^{2n+2}$  such that  $\dot{F}^n(X^{n+L'}, Y^{n+2}, R^{2n+2}) = Z^n$ . To this end, we define the *partial edit probability* depending on the distribution of symbols in the clock-control string  $C^{n+1}$ .

For any  $0 \leq s \leq n$ , a pair  $(l_1, l_2)$  is said to be *permissible* if  $0 \leq l_1, l_2 \leq s+1$  and  $l_1 + l_2 \leq s+1$ . For a given  $s$ , the set of all the permissible values of  $(l_1, l_2)$  is denoted by  ${}_s$ . For any  $1 \leq s \leq n$  and  $(l_1, l_2) \in {}_s$ , the partial edit probability is defined as the conditional joint probability

$$P(l_1, l_2, s) = \Pr\{\dot{O}^s = Z^s, w_1(C^{s+1}) = l_1, w_2(C^{s+1}) = l_2 | X^{s+L'}\} \quad (3)$$

where  $\dot{O}^s = \dot{G}^s(X^{s+L'}, Y^{s+2}, C^{s+1})$  and  $C^{s+1} = H^{s+1}(X^{s+L'}, R^{2s+2})$ . The following theorem shows how to compute the edit probability efficiently, on the basis of a recursive property of the partial edit probability.

**Theorem 1.** *For any given  $X^{n+L'}$  and  $Z^n$ , we have*

$$P(X^{n+L'}; Z^n) = \sum_{(l_1, l_2) \in {}_n} P(l_1, l_2, n) \quad (4)$$

where the partial edit probability  $P(l_1, l_2, n)$  is computed recursively by

$$\begin{aligned}
P(l_1, l_2, s) &= P(l_1 - 1, l_2, s - 1)(1 - z_s - \dot{x}_{s+1-l_2})(1 - x_{L+s-l_2})x_{L+s-l_2-1} \\
&\quad + \frac{1}{8}P(l_1, l_2 - 1, s - 1)(1 - (1 - x_{L+s-l_2+1})x_{L+s-l_2}) \\
&\quad + \frac{3}{8}P(l_1, l_2, s - 1)(1 - (1 - x_{L+s-l_2})x_{L+s-l_2-1})
\end{aligned} \tag{5}$$

for  $1 - s - n$  and all  $(l_1, l_2) -_s$ , with the initial values  $P(0, 0, 0) = \frac{3}{4}(1 - (1 - x_L)x_{L-1})$ ,  $P(0, 1, 0) = \frac{1}{4}(1 - (1 - x_L)x_{L-1})$  and  $P(1, 0, 0) = (1 - x_L)x_{L-1}$ . (For each  $0 - s - n$ , if  $(l_1, l_2)$  is not permissible, then it is assumed that  $P(l_1, l_2, s) = 0$ , so that the corresponding terms in (5) are not computed.)

**Proof** First observe that (4) is a direct consequence of (3) and (1).

Assume that  $s - 2$ . We partition all clock-control strings  $C^{s+1}$  into three subsets with respect to the value of the last symbol  $c_{s+1}$ . For simplicity of notation, the conditioning on  $X^{s+L'}$  is removed from the probability (3) and all the resulting equations. Then (3) can be put into the form

$$\begin{aligned}
P(l_1, l_2, s) &= \\
&\Pr-\dot{o}_s = z_s-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1 - 1, w_2(C^s) = l_2, c_{s+1} = -1- \\
&\quad -\Pr-c_{s+1} = -1-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1 - 1, w_2(C^s) = l_2- \\
&\quad -\Pr-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1 - 1, w_2(C^s) = l_2- \\
&+ \Pr-\dot{o}_s = z_s-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - 1, c_{s+1} = -2- \\
&\quad -\Pr-c_{s+1} = -2-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - 1- \\
&\quad -\Pr-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - 1- \\
&+ \Pr-\dot{o}_s = z_s-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2, c_{s+1} = -1, 2- \\
&\quad -\Pr-c_{s+1} = -1, 2-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2- \\
&\quad -\Pr-\dot{O}^{s-1} = Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2-.
\end{aligned} \tag{6}$$

The third factor in each addend of (6) is easily recognized to be the partial edit probability, of argument  $s - 1$ , appearing in the corresponding addend of (5).

Now, under the condition that  $w_1(C^{s+1}) = l_1 - 1$  and  $w_2(C^{s+1}) = l_2$ , we have  $c_{s+1} = h(x_{L+s-l_2}, x_{L+s-l_2-1}, r_{2s+2}, r_{2s+1})$ , which produces  $-1-$  if and only if  $(x_{L+s-l_2}, x_{L+s-l_2-1}) = (0, 1)$ . Moreover, if  $c_{s+1} = -1-$ , then  $\dot{o}_s = \dot{x}_{s+1-l_2}$ . Similarly, under the condition that  $w_1(C^{s+1}) = l_1$  and  $w_2(C^{s+1}) = l_2$ , we have  $c_{s+1} = h(x_{L+s-l_2+1}, x_{L+s-l_2}, r_{2s+2}, r_{2s+1})$ , which produces  $-2-$  if and only if  $(x_{L+s-l_2+1}, x_{L+s-l_2}) = (0, 1)$  and  $(r_{2s+2}, r_{2s+1}) = (0, 1)$ . Moreover, if  $c_{s+1} = -2-$ , then  $\dot{o}_s = \dot{y}_{s+1-l_1}$ . Finally, under the condition that  $w_1(C^{s+1}) = l_1$  and  $w_2(C^{s+1}) = l_2$ , we have  $c_{s+1} = h(x_{L+s-l_2+1}, x_{L+s-l_2}, r_{2s+2}, r_{2s+1})$ , which produces  $-1, 2-$  if and only if  $(x_{L+s-l_2}, x_{L+s-l_2-1}) = (0, 1)$  and  $(r_{2s+2}, r_{2s+1}) = (0, 1)$ . Moreover, if  $c_{s+1} = -1, 2-$ , then  $\dot{o}_s = \dot{x}_{s+1-l_2} - \dot{y}_{s+1-l_1}$ .

Consequently, we have (conditioned on  $X^{s+L'}$ ) that

$$\begin{aligned} \Pr\text{-}c_{s+1} = -1 - \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1 - 1, w_2(C^s) = l_2 - \\ &= (1 - x_{L+s-l_2})x_{L+s-l_2-1} \end{aligned} \quad (7)$$

$$\begin{aligned} \Pr\text{-}c_{s+1} = -2 - \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - 1 - \\ &= (1 - (1 - x_{L+s-l_2+1})x_{L+s-l_2}) - 1/4 \end{aligned} \quad (8)$$

$$\begin{aligned} \Pr\text{-}c_{s+1} = -1, 2 - \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - \\ &= (1 - (1 - x_{L+s-l_2})x_{L+s-l_2-1}) - 3/4. \end{aligned} \quad (9)$$

Further, we get

$$\begin{aligned} \Pr\text{-}\dot{o}_s = z_s \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1 - 1, w_2(C^s) = l_2, c_{s+1} = -1 - \\ &= 1 - z_s - \dot{x}_{s+1-l_2} \end{aligned} \quad (10)$$

$$\begin{aligned} \Pr\text{-}\dot{o}_s = z_s \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2 - 1, c_{s+1} = -2 - \\ &= 1/2 \end{aligned} \quad (11)$$

$$\begin{aligned} \Pr\text{-}\dot{o}_s = z_s \dot{O}^{s-1} &= Z^{s-1}, w_1(C^s) = l_1, w_2(C^s) = l_2, c_{s+1} = -1, 2 - \\ &= 1/2. \end{aligned} \quad (12)$$

Equation (11) follows from  $\dot{o}_s = \dot{y}_{s+1-l_1}$  by taking into account that  $y_{s+2-l_1}$  remains to be independent of  $y_{s+1-l_1}$  when conditioned on  $\dot{O}^{s-1} = \dot{G}^{s-1}(X^{s-1+L'}, Y^{s+1}, C^s)$ ,  $w_1(C^s) = l_1$ , and  $w_2(C^s) = l_2 - 1$ , as this condition involves only  $Y^{s+1-l_1}$  (not  $y_{s+2-l_1}$ ). Equation (12) is proved analogously.

Equation (5) is obtained from (6) by plugging in the determined probabilities.

For  $s = 1$ , the edit probability values are directly obtained from (3). When these values are expressed in terms of the unknown initial values by the recursion (5), a system of linear equations is obtained. The initial values are then determined by solving this system. —

The time and space complexities of the recursive algorithm corresponding to Theorem 1 are  $O(n^3)$  and  $O(n^2)$ , respectively. Since the edit probability is exponentially small in the string length, the following normalization turns out to be computationally convenient:  $\bar{P}(X^{n+L'}; Z^n) = 2^{n+1}P(X^{n+L'}; Z^n)$ . It results from the right-hand side of (5) and the initial values being multiplied by 2.

## 4 False Alarm Probability

In order to investigate whether a correlation attack based on the proposed edit probability can be successful, we develop a statistical hypothesis testing model similar to the one introduced in [2]. We start by considering the probability distribution of the edit probability  $P(X^{n+L'}, Z^n)$  under the following two probabilistic hypotheses:

- **H<sub>0</sub>** (*correlated case*):  $X^{n+L'}$  and  $Y^{n+L'}$  are purely random and independent and  $Z^n = \dot{F}^n(X^{n+L'}, Y^{n+L'})$  (that is,  $Z^n = \dot{G}^n(X^{n+2}, Y^{n+2}, C^{n+1})$  and  $C^{n+1} = H^{n+1}(X^{n+L}, Y^{n+L})$ ).
- **H<sub>1</sub>** (*independent case*):  $X^{n+L'}$  and  $Z^n$  are purely random and independent.

This means that when generating the correlated samples, the third and the fourth input bits of the clock-control function  $h$  are taken from the input sequence  $Y$  as in the actual BSG scheme, rather than from an auxiliary random sequence. For the correlation attack to work, it is necessary that the separation between the probability distributions in the correlated and independent cases increases with the string length  $n$ , and the faster the increase, the smaller the string length required for successful decision making is. Analyzing the separation of the two probability distributions seems to be a difficult task from a theoretical point of view, but one can measure the separation experimentally.

We conducted systematic experiments for the normalized edit probability and produced histograms of the two distributions for each  $n = 100, (10), 800$  on random samples of 1000 pairs  $(X^{n+L'}, Z^n)$  generated according to  $\mathbf{H}_0$  (for fixed  $L = 100$ ) and  $\mathbf{H}_1$ , respectively. They show that the separation of interest increases with the string length  $n$ . It thus turns out that, for a sufficiently large  $n$ , the normalized edit probability is much larger in the correlated than in the independent case. For illustration, Tables 1 and 2 given in the Appendix display the observed minimum, maximum, mean, and median values along with the standard deviation of the normalized edit probability, for each  $n = 100, (100), 800$ , for the independent and correlated case, respectively.

As we deal with a decision making problem, the separation between the two distributions is measured by the false alarm probability (derived from the distribution under  $\mathbf{H}_1$ ) when the missing event probability (derived from the distribution under  $\mathbf{H}_0$ ) is fixed. Since the number of correct input strings is only one, reasonable values for the missing event probability seem to be  $p_m = 0.1$  or  $p_m = 0.05$ . Therefore, in the statistical hypothesis testing considered, a threshold is set according to  $p_m$  and a tested input string is classified under  $\mathbf{H}_0$  or  $\mathbf{H}_1$  depending on whether the normalized edit probability is bigger or smaller than the fixed threshold. The false alarm probability  $p_f$  then becomes a function of  $n$ , and if and only if this function is decreasing, the separation between the two distributions increases with  $n$ , as desired. The value of  $n$  should be chosen so as to make  $p_f$  inversely proportional to the number of incorrect input strings. We evaluated thresholds and false alarm probabilities relative to the data collected for the above histograms. For illustration, Table 3 given in the Appendix displays the estimated threshold,  $\bar{P}_{th}$ , and false alarm probability,  $p_f$ , for each  $n = 100, (100), 800$ .

The data collected show that, for each considered  $p_m$ , the estimated  $p_f$  decreases with  $n$ . Moreover, for large  $n$ ,  $p_f$  appears to be following the exponential form  $a b^n$ ,  $b < 1$ . As a consequence, the minimum string length  $n$  required for the expected number of false input string candidates to be reduced to about one is linear in the logarithm (to the base two) of the total number of tested input strings. The corresponding estimates of the parameters  $a$  and  $b$  were obtained by the least mean square approximation method applied to the logarithms to the base two of the false alarm probability estimates for  $n = 100, (10), 800$  and are presented in Table 4 given in the Appendix. The parameters  $a$  and  $b$  were estimated on the first 10, 20, and 25 points for  $p_m = 0.1$  and  $p_m = 0.05$ . The most

reliable estimates were obtained for the first 10 points. To be on the conservative side, the false alarm probabilities  $p_f^{0.1}(n)$  and  $p_f^{0.05}(n)$  can be approximated for large  $n$  by

$$p_f^{0.1}(n) \approx 0.542 - 0.986^n, \quad p_f^{0.05}(n) \approx 0.520 - 0.990^n. \quad (13)$$

## 5 Correlation Attack

In this section, we propose a correlation attack on the BSG based on the properties of the introduced edit probability. It is assumed that the LFSR feedback polynomials and a sufficiently long segment of the BSG output sequence are known to the cryptanalyst. The attack consists of two phases. The goal of the first phase is to recover the initial state of LFSR<sub>1</sub> by using the (normalized) edit probability defined in Section 3. Suppose a number of candidates for the LFSR<sub>1</sub> initial state is obtained in this way. Then, in the second phase, the correct initial states of LFSR<sub>1</sub> and LFSR<sub>2</sub> are reconstructed. The solution is very likely to be unique as the equivalent initial states, producing the same output sequence, are unlikely to exist.

We first produce  $n$  bits of the first binary derivative,  $Z^n$ , of the first  $n + 1$  successive bits of the known BSG output sequence. The string  $Z^n$  comes from the (unknown) output strings of the regularly clocked LFSR<sub>1</sub> and LFSR<sub>2</sub> of maximum possible length  $n + L$  (the actual lengths are random and depend on the unknown LFSR initial states). In the correlation attack on LFSR<sub>1</sub>, for any possible LFSR<sub>1</sub> initial state, by using its linear recursion, we first generate  $n + L$  output bits which constitute the input string  $X^{n+L'}$ . Then, we compute the normalized edit probability  $\bar{P}(X^{n+L'}, Z^n)$  by the recursive algorithm derived in Section 3. This is repeated for the  $2^L - 1$  possible initial states of LFSR<sub>1</sub> (the all zero state is excluded). Roughly speaking, the candidates for the correct LFSR<sub>1</sub> initial state are obtained as the ones with the computed normalized edit probability close to being maximal. More precisely, a threshold is set according to the missing event probability (for the correct hypothesis  $\mathbf{H}_0$ ) which is fixed to a value which need not be very small (e.g.,  $p_m = 0.1$  or  $p_m = 0.05$ ). Then, every possible initial state is classified as a candidate if the corresponding normalized edit probability is not less than the threshold. This threshold can be obtained experimentally, as described in Section 4.

Ideally, for  $n$  sufficiently large, there should remain only one candidate for the initial state of LFSR<sub>1</sub>. This can happen if and only if the false alarm probability (for the alternative hypothesis  $\mathbf{H}_1$ )  $p_f(n)$  defined in Section 4 is sufficiently small. Namely, since the expected number of false candidates for an average  $Z^n$  is  $(2^L - 1)p_f(n)$ , the correlation attack would be successful if and only if, approximately,

$$2^L p_f(n) \approx 1. \quad (14)$$

If  $p_f(n)$  has the exponential form  $a b^n$ , where  $b < 1$ , then (14) reduces to

$$n \approx \frac{L + \log_2 a}{-\log_2 b} \quad (15)$$

which means that the required output segment length is linear in the length of LFSR<sub>1</sub>. By (13), the required output segment length is then estimated as

$$n - 49.2L - 43.4 \quad (16)$$

$$n - 69.0L - 65.1 \quad (17)$$

for  $p_m = 0.1$  and  $p_m = 0.05$ , respectively.

Accordingly, we should obtain a relatively small number of candidate initial states for LFSR<sub>1</sub> in time  $O(2^{L+3\log_2 L})$ . These candidate initial states are then ranked in order of decreasing normalized edit probabilities. The candidate states are tested in the second phase of the attack, according to decreasing normalized edit probabilities. Namely, each candidate state is associated with each of the  $2^L - 1$  possible initial states for LFSR<sub>2</sub>. Each resulting LFSR state pair is then used to initialize the state of the BSG under attack. The corresponding output sequence is finally compared with the given BSG output sequence. All the solutions for the LFSR initial states are thus found in time  $O(2^L)$ .

## 6 Experimental Results

The attack described in the previous section was tested on short LFSRs by computer simulations, which verified that the attack can work in practice.

Some results of our experiments are shown in Table 5. In each experiment we used primitive LFSRs of the same length  $L$ . For  $L = 14, 15$ , and  $16$ , the needed BSG output string length,  $n + 1$ , was first estimated by (16). The experiments were also repeated by halving this string length. The good results (and time reduction) obtained for  $L = 14, 15$ , and  $16$  when moving from  $n$  to  $n/2$  motivated the choice of using  $n = 500(400)$  instead of  $n = 800(400)$ , for  $L = 17$ . The thresholds for the normalized edit probability were obtained from the data collected for  $n = 100, (10), 800$  (see Table 3 for  $n = 100, (100), 800$ ). For  $n = 325, 375$ , we used interpolation.

We counted the number of LFSR<sub>1</sub> states giving rise to a normalized edit probability not smaller than the given threshold (*candidates*). For every candidate initial state for LFSR<sub>1</sub> we searched for a companion initial state for LFSR<sub>2</sub> and counted the joint solutions (*solutions*). Table 5 shows that a unique joint solution was always found. Finally, we determined the position of the LFSR<sub>1</sub> component of this solution in the list of LFSR<sub>1</sub> initial state candidates ranked in order of decreasing normalized edit probabilities (*rank*).

As for the LFSR<sub>1</sub> initial state candidates, we found that a candidate is very likely obtainable from the correct LFSR<sub>1</sub> initial state by a small positive or negative phase shift.

## 7 Conclusions

It is pointed out that the stop/go clocking in the bilateral stop/go keystream generator can be viewed as a random edit transformation of an input string into

one output binary string. The input string corresponds to the output sequence of LFSR<sub>1</sub> when regularly clocked and the output string corresponds to the first binary derivative of the keystream sequence. The output sequence of LFSR<sub>2</sub> and an auxiliary binary clock-control string are assumed to be independent and purely random. The related edit probability is then defined and a recursive algorithm for its computation is derived.

It is shown how the edit probability can be used to mount a correlation attack on LFSR<sub>1</sub>. For the underlying statistical hypothesis testing problem, the false alarm probability is estimated by computer simulations. According to the experiments conducted, the minimum output sequence length required to be known for a successful attack is linear in the length,  $L$ , of the LFSRs. The time complexity of the attack is estimated as  $O(2^{L+3\log_2 L})$ . Successful experimental correlation attacks performed on relatively short shift registers demonstrate the effectiveness of the developed methodology.

## References

1. D. Gollmann and W. G. Chambers, "Clock-controlled shift registers: A review," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 525-533, May 1989.
2. J. Dj. Golić and R. Menicocci, "Edit probability correlation attack on the alternating step generator," *Sequences and Their Applications - SETA '98, Discrete Mathematics and Theoretical Computer Sciences*, C. Ding, T. Helleseht, and H. Niederreiter eds., Springer-Verlag, pp. 213-227, 1999.
3. K. Zeng, C. H. Yang, and T. R. N. Rao, "Large primes in stream-cipher cryptography," *Advances in Cryptology - AUSCRYPT '90, Lecture Notes in Computer Science*, vol. 453, J. Seberry and J. Pieprzyk eds., Springer-Verlag, pp. 194-205, 1990.
4. K. Zeng, C. H. Yang, D. Y. Wey, and T. R. N. Rao, "Pseudorandom bit generators in stream-cipher cryptography," *IEEE Computer*, vol. 24, no. 2, pp. 8-17, Feb. 1991.

## Appendix

<b>Table 1.</b> Statistics of $\bar{P}$ on 1000 independent pairs $(X^{n+L'}, Z^n)$ .					
$n$	Min	Max	Mean	Median	Std Dev
100	6.341E-9	3.339E2	2.314E0	1.43E-1	1.294E1
200	4.307E-15	3.952E2	2.121E0	1.597E-2	1.69E1
300	4.279E-15	2.774E2	1.632E0	2.674E-3	1.347E1
400	2.669E-14	6.531E2	1.2E0	4.36E-4	2.168E1
500	2.246E-14	7.893E1	2.456E-1	1.13E-4	2.82E0
600	6.1E-20	3.019E2	4.643E-1	2.023E-5	9.68E0
700	3.915E-18	1.598E4	1.613E1	3.771E-6	5.055E2
800	5.145E-19	7.301E1	1.669E-1	7.699E-7	2.848E0

Table 2. Statistics of $\bar{P}$ on 1000 correlated pairs $(X^{n+L'}, Z^n)$ .					
$n$	Min	Max	Mean	Median	Std Dev
100	7.637E-2	7.097E3	1.097E2	2.486E1	4.006E2
200	8.051E-2	1.703E6	6.538E3	2.033E2	7.162E4
300	5.31E-2	3.59E7	1.269E5	1.863E3	1.62E6
400	3E-1	2.658E8	1.759E6	1.056E4	1.438E7
500	3.053E0	7.872E10	1.213E8	1.108E5	2.573E9
600	8.349E-1	3.3E12	6.27E9	6.357E5	1.222E11
700	4.942E1	2.123E12	4.933E9	5.135E6	7.462E10
800	3.484E0	1.987E13	7.507E10	2.374E7	8.822E11

Table 3. Estimation of thresholds and false alarm probabilities.				
$n$	$\bar{P}_{th}^{0.1}$	$\bar{P}_{th}^{0.05}$	$p_f^{0.1}$	$p_f^{0.05}$
100	3.122E0	1.893E0	1.22E-1	1.67E-1
200	1.214E1	5.655E0	2.7E-2	4.8E-2
300	4.138E1	1.687E1	1.1E-2	1.9E-2
400	1.808E2	5.709E1	1E-3	3E-3
500	7.413E2	2.822E2	0	0
600	7.164E3	1.505E3	0	0
700	2.172E4	3.835E3	0	1E-3
800	7.587E4	1.891E4	0	0

Table 4. Estimation of $a$ and $b$ on 10 , 20 , and 25 points.						
$p_m$	$a$	$a$	$a$	$b$	$b$	$b$
0.1	.542	.585	.551	.986	.986	.986
0.05	.520	.736	.731	.990	.987	.987

Table 5. Experimental results.					
$L$	$n$	$threshold$	$candidates$	$solutions$	$rank$
14	650(325)	11290(75)	20(51)	1(1)	1(1)
15	700(350)	21720(109)	8(72)	1(1)	1(5)
16	750(375)	46070(145)	30(171)	1(1)	1(1)
17	500(400)	741(180)	35(139)	1(1)	26(29)

# Look-Up Table Based Large Finite Field Multiplication in Memory Constrained Cryptosystems

(Extended Abstract)

**Abstract.**  $n$   $l$   $l$   $k$   $l$   $l$   $n$

$l$   $l$   $n$   $n$   $n$   $n$   $l$

$n$   $n$   $n$   $n$   $n$   $l$   $k$   $l$

$l$   $n$   $n$   $l$   $n$

$l$   $n$   $n$   $l$   $l$   $l$   $n$   $n$   $n$

$n$   $n$   $n$   $n$   $n$   $n$

# 1 Introduction

$l$   $l$   $l$   $l$   $m$   $n$   $m$   $l$   $l$   
 $g$   $m$   $l$   $l$   $a$   $b$   $n$   $l$   $m$   
 $l$   $l$   $m$   $m$   $m$   $l$   $l$   $m$   $n/g$   
 $l$   $l$   $m$   $m$   $m$   $l$   $l$   $ll$   $l$   
 $l$   $m$   $m$   $l$   $m$   $l$   $l$   
 $m$   $l$   $l$   $m$   $l$   $m$   $l$   
 $l$   $m$   $l$   $l$   $m$   $m$   $l$   $l$   
 $m$   $l$   $m$   $l$   $m$   $l$   $m$   $l$   $m$   
 $l$   $m$   $l$   $m$   $m$   $m$   $l$   $m$   $l$   $l$   
 $l$   $m$   $m$   $l$   $m$   $m$   $m$   $l$   $m$   
 $l$   $l$   $l$   $m$   $l$   $l$   $l$   $l$   $l$   
 $l$   $l$   $l$   $m$   $n$   $m$   $l$   $l$   $l$   
 $l$   $l$   $l$   $l$   $m$   $l$   $l$   $l$   $m$   
 $l$   $l$   $l$   $l$   $l$   $l$   $l$   $m$   
 $l$   $ll$   $l$   $m$   $m$



$$k \qquad l \qquad n \qquad l \qquad l \qquad l \qquad n$$

$$\begin{aligned} P\ x \quad & A\ x\ B\ x\ m \quad F\ x \\ & A\ x\ (b_{n-1}x^{n-1} \quad b_{n-2}x^{n-2} \quad \text{---} \quad b_1x \quad b_0) \ m \quad F\ x \\ & \text{---} \quad A\ x\ b_{n-1}x \quad A\ x\ b_{n-2}\ x \quad \text{---} \quad x \quad A\ x\ b_1\ x \quad A\ x\ b_0\ m \quad F\ x \\ & \text{---} \quad A\ x\ b_{n-1}x\ m \quad F\ x \quad A\ x\ b_{n-2}\ x\ m \quad F\ x \\ & \text{---} \quad x\ m \quad F\ x \quad A\ x\ b_1\ x\ m \quad F\ x \quad A\ x\ b_0. \\ & \qquad \qquad \qquad l \quad m\ l \quad l \quad A\ x \qquad x \\ & \qquad \qquad \qquad B\ x \qquad \qquad \qquad m \quad F\ x \\ & \qquad \qquad \qquad m \qquad l \qquad m \qquad ll \end{aligned}$$

$$\begin{aligned} \textbf{Algorithm 1} \qquad & l \quad l \quad m \qquad n \quad l \quad l \\ & A\ x\ B\ x \quad F\ x \\ & P\ x \quad A\ x\ B\ x\ m \quad F\ x \end{aligned}$$

$$\begin{aligned} \textbf{Step 1.1} \quad & b_{n-1} \quad P\ x \quad A\ x \\ & l \quad P\ x \end{aligned}$$

$$\begin{aligned} \textbf{Step 1.2} \quad & i \quad n - \quad - \\ & P\ x \quad xP\ x\ m \quad F\ x \\ & b_i \quad P\ x \quad P\ x \quad A\ x \end{aligned}$$

$$\begin{aligned} & \qquad \qquad \qquad l \qquad P\ x \qquad l \quad m\ l \\ & \qquad \qquad \qquad a-b \quad l \quad m \qquad \qquad \qquad i.e. \quad l \qquad B\ x \\ & \qquad \qquad \qquad l \qquad \qquad \qquad n - \quad m \\ & P\ x \quad xP\ x\ m \quad F\ x \qquad l \quad l \\ & P\ x \quad xP\ x \qquad \qquad \qquad F\ x \quad m \qquad n \\ & \qquad \qquad \qquad xP\ x \quad m \\ & P\ x \quad l \quad m\ l \qquad \qquad \qquad P\ x \quad xP\ x\ m \quad F\ x \\ & \frac{n-1}{2} \quad l \quad m\ l \qquad \qquad \qquad m\ l \quad l \quad \frac{n-1}{2} \quad m \quad l \quad m\ l \\ & \qquad \qquad \qquad P\ x \quad P\ x \quad A\ x \\ & l \quad m \qquad \qquad \qquad m \quad l \quad n - \quad l \quad m\ l \\ & \qquad \qquad \qquad m \quad m \qquad \qquad \qquad l \quad m\ l \quad T_{poly\_add} \\ & m \quad m \qquad \qquad \qquad n \quad m\ l \quad l \qquad \qquad \qquad m \\ & \qquad \qquad \qquad T_{multi\_in\_GF(2^n)} \approx n - \quad -T_{poly\_add}. \\ & \qquad \qquad \qquad m \qquad \qquad \qquad l \quad l \quad n \qquad \qquad \qquad l \quad l \quad l \quad m \\ & m \qquad \qquad \qquad l \qquad \qquad \qquad ll \qquad \qquad \qquad m \end{aligned}$$

### 3 Group-Level Look-Up Table Based Multiplication

$$\begin{aligned} & \qquad \qquad \qquad n \qquad \qquad \qquad B\ x \quad s \qquad \qquad \qquad g - \qquad 1 \\ & n \qquad \qquad \qquad m\ l \quad l \quad g \qquad \qquad \qquad m \qquad \qquad \qquad m \\ & \qquad \qquad \qquad n\ m \quad g \\ & \hline & 1 \quad nl\ k \qquad \qquad \qquad g \quad n \qquad \qquad \qquad k \qquad \qquad \qquad ll \\ & \qquad \qquad \qquad l \qquad \qquad \qquad n \quad n \quad n \qquad \qquad \qquad l \quad n \quad n \quad l \quad g \quad l \quad w \end{aligned}$$



$$\begin{aligned}
& \text{K} \quad \text{L} \quad \text{N} \quad \text{L} \quad \text{L} \quad \text{L} \quad \text{N} \\
& \text{F} \text{ x} \quad \text{F} \text{ x} \quad \text{L} \\
& \text{m} \quad \text{m} \quad \text{m} \\
& \text{L} \quad \text{M} \quad \text{m} \quad \text{L} \quad \text{L} \quad \text{m} \quad \text{L} \quad \text{x}^g \left( \sum_{i=n-g}^{n-1} p_i x^i \right) \text{m} \quad \text{F} \text{ x} \\
& \text{L} \quad \text{P} \text{ x} \quad \text{P} \text{ x} \quad \text{B}_k \text{ x} \text{ A} \text{ x} \text{ m} \quad \text{F} \text{ x} \\
& \text{L} \quad \text{X}_3 \quad \text{B}_k \text{ x} \text{ A} \text{ x} \text{ m} \quad \text{F} \text{ x} . \quad 9 \\
& \text{B}_k \text{ x} \text{ A} \text{ x} \quad \text{L} \quad \text{L} \quad \text{m} \quad \text{L} \quad - \quad n-g- \\
& \text{x}^{n-g-2}, \text{x}^{n-g-3}, \text{---}, \text{x}^n \quad \text{B}_k \text{ x} \text{ A} \text{ x} \quad \text{L} \\
& \text{m} \text{ L} \quad \text{L} \quad \text{m} \quad \text{m} \quad \text{L} \quad \text{L} \quad \text{g} \quad \text{m} \text{ X}_3 \quad \text{T} \\
& n > g \quad \text{L} \quad \text{m} \quad \text{L} \quad \text{L} \quad \text{B}_k \text{ x} \text{ A} \text{ x} \text{ m} \quad \text{F} \text{ x} \\
& \text{L} \quad \text{L} \quad \text{B}_k \text{ x} \quad \text{L} \quad \text{L} \quad \text{L} \quad \text{n}^g \\
& \text{L} \quad \text{A} \text{ x} \quad \text{M} \quad \text{L} \quad \text{L} \quad \text{fl} \quad \text{m} \\
& \text{L} \quad \text{L} \quad \text{m} \quad \text{L} \quad \text{L} \quad \text{L} \quad \text{m} \\
& \text{L} \quad \text{L} \quad \text{m} \quad \text{L} \quad \text{L} \quad \text{L} \quad \text{m} \\
& \text{L} \quad \text{e}^{\Delta} \left( \sum_{i=0}^{g-1} e_i x^i \right) \quad \text{M} \quad \text{T} \quad \text{L} \quad \text{L} \quad \text{m} \\
& \text{e} \quad \text{M} \quad \text{T} \quad \text{L} \quad \text{L} \quad \text{m} \\
& \text{M} \text{ e} \quad e_{g-1} x^{g-1} \quad e_{g-2} x^{g-2} \quad \text{---} \quad e_0 x^n \text{ m} \quad \text{F} \text{ x} , \\
& \text{T} \text{ e} \quad e_{g-1} x^{g-1} \quad e_{g-2} x^{g-2} \quad \text{---} \quad e_0 \text{ A} \text{ x} \text{ m} \quad \text{F} \text{ x} , \\
& \text{L} \quad \text{L} \quad \text{L} \quad \text{m} \\
& P_{s-1} \text{ x}^{\Delta} \left( \sum_{i=0}^{g-1} p_{n-g+i} x^i \right)
\end{aligned}$$

**Algorithm 3**  $\text{L} \quad \text{L} \quad \text{L} \quad \text{L}$

$$\begin{aligned}
& \text{A} \text{ x} \text{ B} \text{ x} \text{ F} \text{ x} \quad \text{M} \quad \text{L} \\
& \text{P} \text{ x} \quad \text{A} \text{ x} \text{ B} \text{ x} \text{ m} \quad \text{F} \text{ x}
\end{aligned}$$

- Step 3.1**  $\text{L} \text{ T}$
- Step 3.2**  $\text{P} \text{ x} \quad \text{T} \text{ B}_{s-1} \text{ x}$
- Step 3.3**  $\text{k} \quad \text{s} - \quad -$
- $$\begin{aligned}
& \text{X}_1 \quad \text{x}^g \left( \sum_{i=0}^{n-1-g} p_i x^i \right) \\
& \text{X}_2 \quad \text{M} \text{ P}_{s-1} \text{ x} \\
& \text{X}_3 \quad \text{T} \text{ B}_k \text{ x} \\
& \text{P} \text{ x} \quad \text{X}_1 \text{ X}_2 \text{ X}_3
\end{aligned}$$

$$\begin{aligned}
& \text{L} \quad \text{m} \quad \text{m} \quad \text{n} \quad \text{m} \quad \text{T} \quad \text{M} \quad \text{L} \\
& \text{s} \quad \text{s} - \quad \text{L} \quad \text{L} \quad \text{m} \quad \text{s} - \quad \text{L} \quad \text{m} \quad \text{L} \\
& \text{s} - \quad \frac{n}{w} - \quad \text{L} \quad \text{X}_1 \quad \text{s} - \quad \frac{n}{w} - \\
& \text{s} - \quad \text{L} \quad \text{s} - \\
& \text{s} - \quad , \quad \text{---} \quad , \quad \text{L} \quad \text{m} \quad \text{T} \text{ B}_i \text{ x} \quad i
\end{aligned}$$

#### 4.1 Entry Computation on Demand

**Algorithm 4**

$$k \qquad l \qquad n \qquad l \qquad l \qquad l \qquad n$$

$$\textbf{Step 4.1} \quad e_0 \quad tmp$$

$$l \quad tmp \quad T$$

$$\textbf{Step 4.2} \quad g - \quad -$$

$$e_i \quad tmp \quad tmp \quad T^i$$

$$-$$

$$\textbf{Step 4.3} \quad T e \quad tmp \quad -$$

$$m \qquad l \quad m \quad l \qquad l \qquad m$$

$$mm \qquad e \qquad l \quad m$$

$$g - \quad / \quad l \quad m \quad l \qquad m \qquad T B_i x$$

$$- i - s - \qquad l \quad m \qquad m \quad s g - \quad / \quad l \quad m \quad l$$

$$\qquad ll \qquad l \qquad T \qquad ll$$

$$m \qquad l$$

$$\textbf{Corollary 2.} \qquad ll \qquad l \qquad T \quad l \quad -$$

$$l \quad m \qquad g^{-1} g - \qquad l \quad m \quad l$$

# 4.2 Entry Computation in Window Sequence

$$s - g \qquad l$$

$$l \qquad ll \qquad l$$

$$g - g - \qquad l \qquad g - \qquad m \quad l$$

$$W_1, W_2, \text{---}, W_{g-1} \qquad W_i \quad - i - g - \qquad ll$$

$$i - \qquad T^i \qquad , T^i \qquad , \text{---}, T^i \quad i -$$

$$\textbf{Lemma 3.} \qquad W_i$$

$$T^i \quad j \quad T^i \quad T j, \quad - j - i - .$$

$$- j - i \qquad m \qquad W_j \quad - j - i - \qquad T^j$$

$$m \qquad W_i \qquad l \qquad l \quad m \quad l$$

$$m \qquad l \qquad m \qquad i.e. \qquad W_i$$

$$W_{i-1} \qquad m \qquad m \qquad ll$$

$$l \qquad m \qquad l$$

$$\textbf{Algorithm 5} \quad m \qquad l$$

$$T^i, i \quad , \text{---}, g -$$

$$ll \quad l \qquad T$$

$$i \qquad g - \quad -$$

$$j \quad i-1 - \quad -$$

$$T^i \quad j \quad T^i \quad T j$$

$$\begin{array}{ccccccc}
 & & n & & & & \\
 & l & m & l & i & m & \\
 & & i & - & W_i & & ll \\
 g - & & & & & & \\
 & 1 & - & 2 & - & g-1 & - \\
 & & & & & g-1 & - \\
 & & & & & & - g \\
 l & m & l & & & & 
 \end{array}$$

## 5 Conclusions

$$\begin{array}{ccccccc}
 & & n & m & l & l & l \\
 & s & n & & & & l \\
 s/ & g & m & l & m & l & \\
 & l & & l & g & \frac{n}{w} & - \\
 m & l & m & ll & m & n & l \\
 l & & g & & m & m & \\
 & m & l & l & l & m & l \\
 & & & & & & m
 \end{array}$$

## Acknowledgment

$$\begin{array}{ccccccc}
 & & & & l & m & \\
 & ll & m & & l & l & \\
 m & & & l & l & l & m \\
 & l & l & m & & & \\
 l & & & & & & 
 \end{array}$$

## References

*VLSI Architectures for Computations in Galois Fields*  
 l n nk n n nk n n  
 n n l l ll n l *IEEE Trans.*  
*Computers*  
 n ll l l l l  
 m *Inform. and Comp.* l  
 n n n l n n l  
 l ll l l l n l m *IEEE Trans.*  
*Comput.* l  
 n n n n l k ll  
 k l n n *Advances in Cryptology- EUROCRYPT '92, Lecture Notes in Com-*  
*puter Science* n l  
 n l n n n n ll  
 l n n n n n *Advances*  
*in Cryptology- ASIACRYPT '96, Lecture Notes in Computer Science*  
 n

k l n l l l n  
 n n l l n n <sup>k</sup> *Design, Codes and*  
*Cryptography* l  
 n n l ll  
 n *Advances in Cryptology- CRYPTO '97, Lecture Notes in Computer Science*  
 n l  
 ll l n l l l  
 l n l *IEEE Trans. Computers* l  
 n ll n  
 n *Standards for Efficient Cryptography Group*



## 1.2 Pseudoprimality Testing Based on the Lucas Sequences

**Definition 1** Let  $(L_n)$  be the Lucas sequence of the first kind and let  $m$  be any integer. The rank of appearance modulo  $m$  (or simply  $r(m)$ ) is the smallest integer  $n$ , if it exists, such that  $L_n \equiv 0 \pmod{m}$ .

$$n^{n-1} \mid n - \frac{D}{p} \left( n - \left( \frac{D}{n} \right) n \right) = 0$$

**Lemma 1** *Let  $n$  be a positive, odd composite integer and  $\chi \in \mathbb{Z}_n^*$ . Then  $\chi$  is a  $\text{sLpsp}$  if and only if it is a  $\text{Lpsp}$  and  $\chi_2$  is a constant value for all prime factors of  $n$ .*

sLpsp      n      n  
n      11      n

### 1.3 Combined Tests

n n n n n n n n n n n n n n n





**Remark 1** Note that  $\mathcal{N}_0$  counts the values of  $\mathbf{v}$  with both  $\left(\frac{D(P)}{P}\right) \left(\frac{D(Q)}{P}\right) = 1$  and  $\mathbf{v} \in \mathcal{V}_0$  if  $\mathbf{v} \in \mathcal{V}_0$  and  $\mathbf{v} \in \mathcal{V}_0$  if  $\mathbf{v} \in \mathcal{V}_0$ .

$$n \qquad n \qquad n \quad 1$$

**Proposition 1** *Let  $\mathcal{C}$  be a curve of genus  $g$  and suppose  $\mathcal{C}$  is a divisor of  $\mathcal{C}$  where  $\mathcal{C} = -1 - 1 -$*

- $$\begin{aligned} 1. \text{ If } \left( \frac{Q_0}{p} \right) &= -1 \text{ then } \begin{cases} 0 & \text{if } 2 \mid n-1 \\ 0 & \text{otherwise.} \end{cases} \\ 2. \text{ If } \left( \frac{Q_0}{p} \right) &= 1 \text{ then } \begin{cases} 0 & \text{if } 2 \mid n-1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

### 3 The Number of $P$ That Pass the Lucas Test for a Fixed $Q = Q_0$

**Proposition 2** *For any odd prime  $p$  and any positive integers  $a$  and  $b$  the following is true.*

$$\begin{aligned} & \text{if either } \begin{matrix} k & -0 \\ 2k & -0 \end{matrix} \text{ or } \begin{matrix} 2^i k & -0 \\ k & -0 \end{matrix} \text{ then } \begin{matrix} 2^i k & -0 \\ k & -0 \end{matrix} \\ & \text{iff either } \begin{matrix} 2^i k & -0 \\ k & -0 \end{matrix} \text{ or } \begin{matrix} k & -0 \\ 2k & -0 \end{matrix} \end{aligned}$$

*Proof.*

**Theorem 1** Let  $p$  be an odd prime,  $a, b$  positive integers,  $c \geq 1$ , and  $d \geq -1$  a constant. For a fixed value of  $a, b, c, d$ , the number of distinct numbers  $n \leq x$  with  $\left(\frac{D(P)}{P}\right)^a \left(\frac{D(Q)}{Q}\right)^b \leq n \leq x$  and  $k \leq n \leq x$  is given in the following way.

- $$\begin{aligned} 1. \text{ For } \left(\frac{Q_0}{p}\right) & \begin{cases} -1 & \text{as } \left(\frac{(k,p-\epsilon)}{2}\right) \text{ when } 2 \mid k-2 \\ 0 & \text{otherwise,} \end{cases} \\ 2. \text{ for } \left(\frac{Q_0}{p}\right) & \begin{cases} 1 & \text{as } \left(\frac{(k,p-\epsilon)}{2}\right) - 1 \text{ when } 2 \mid k-2 \\ -1 & \text{otherwise.} \end{cases} \end{aligned}$$

*Proof.*

$$- \frac{k}{n} \frac{n}{0} - 0 \frac{n}{n} \frac{n}{n} \frac{n}{n} \frac{n}{0} \mid \frac{n}{0} \frac{n}{0} \frac{\alpha}{0} 1$$

[illegible]

**Theorem 2** *Under the hypotheses of Theorem 1, the number of parameters with  $k_0 = 0$  is  $O(p^\alpha)$  for a fixed  $0 \leq \alpha$  is,*

$$\begin{aligned} & - \text{when } \left( \frac{Q_0}{p} \right) \begin{pmatrix} -1 & \text{given as } 0 \\ \frac{p-\epsilon}{2} & \text{otherwise} \end{pmatrix} \quad \begin{matrix} \text{for } 2 & 1 & 2 & - \\ & & & \end{matrix} \\ & - \text{when } \left( \frac{Q_0}{p} \right) \begin{pmatrix} 1 & \text{given as } 0 \\ - & \text{otherwise.} \end{pmatrix} \quad \begin{matrix} \text{for } 2 & 1 & 2 & - \\ & & & \end{matrix} \end{aligned}$$

*Proof.*

$$\begin{aligned} & \alpha_n \quad n \quad k \quad 0 \quad -0 \quad \alpha_n \quad \left( \frac{Q_0}{p} \right) \quad \left( \begin{matrix} 2k & 0 & -0 \\ & -1 & \end{matrix} \right) \\ & n \quad 1 \quad n \quad 2 \quad - \quad 2 \quad - \quad \frac{(2k, p-\epsilon)}{2} - \frac{(k, p-\epsilon)}{2} \\ & 0 \quad 2 \quad 2 \quad 1 \quad 2 \quad - \quad n \quad \frac{(2k, p-\epsilon)}{2} \quad \frac{p-\epsilon}{2} \quad n \\ & 2 \quad 1 \quad 2 \quad - \quad n \quad 0-0 \quad n \quad nn \\ & \left( \frac{Q_0}{p} \right) \quad 1 \end{aligned} \quad \square$$

## 4 The Proposed Strong Fermat/Strong Lucas Combination

### 4.1 Some Fundamentals

$$\begin{aligned} & \text{Epsp} \quad \frac{n-1}{2} - \left( \frac{Q}{n} \right) \\ & n \quad 1 \quad (n-(D/n))/2 - 0 \quad (n-(D/n))/2 - 0 \\ & n \quad \left( \frac{Q}{n} \right) \quad 1 \quad -1 \quad n \quad n \quad 1 \\ & n \quad n \quad n \quad n \quad n \\ & \text{ELpsp} \quad \left( \frac{Q}{n} \right) \quad -1 \\ & \text{spsp} \quad 1 \quad n \quad \text{ELpsp} \quad n \quad \left( \frac{Q}{n} \right) \quad -1 \\ & - \quad - \quad n \quad \text{sLpsp} \end{aligned}$$

### 4.2 Description of the Proposed Test

The test for one :

$$\begin{aligned} & 1 \quad 0 - \mathbb{Z}_n \quad \left( \frac{Q_0}{n} \right) \quad -1 \\ & n \quad n \quad \text{Epsp} \quad 0 \quad n \quad n \\ & -\mathbb{Z}_n \quad -0 \quad 2 - 0 \quad n \quad \left( \frac{D(P)}{n} \right) \quad -1 \\ & \text{ELpsp} \quad 0 \quad \frac{n+1}{2} \quad 0 \quad -0 \quad n \quad n \quad n \\ & n \quad n \quad n \quad \text{spsp} \quad 0 \quad n \quad \text{sLpsp} \quad 0 \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \\ & n \quad i \quad n \quad n \quad n \quad n \quad n \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \end{aligned}$$

In the following, we will keep  $0$  fixed and check condition 4 for different choices of  $i$ .

$$\begin{aligned} & n \quad i \quad n \quad n \quad \text{sLpsp} \quad i \quad 0 \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \\ & n \quad n \quad n \quad n \quad n \quad n \quad n \end{aligned}$$



– if  $\left(\frac{Q_0}{p} \mid -1\right)$  then  $\left(\frac{Q_0}{n} \mid -1\right)$  and  $\left(\frac{Q_0}{n} \mid -1\right)$

*Proof.*  $n$  □

## 6 The Number of Parameters That Pass the Proposed Test

### 6.1 Some Technical Prerequisites

**Lemma 2** If  $n$  is a sLpsp such that  $\left(\frac{Q_0}{n} \mid -1\right)$  then  $\left(\frac{D(P)}{n} \mid -1\right)$  for all prime divisors  $p$  of  $n$ .

*Proof.*  $n$  is a ELpsp  $\left(\frac{D(P)}{n} \mid -1\right)$  □

**Lemma 3** A necessary condition for an odd composite number  $n$  to simultaneously pass a psp  $\left(\frac{D(P)}{n} \mid -1\right)$  and a Lpsp  $\left(\frac{D(P)}{n} \mid -1\right)$  is that  $n \equiv 0 \pmod{4}$

*Proof.*  $n \equiv 0 \pmod{4}$  □

### 6.2 The Main Results

$n$  is a psp  $\left(\frac{D(P)}{n} \mid -1\right)$  if and only if  $n \equiv 0 \pmod{4}$

**Proposition 3** Let  $n$  be a psp  $\left(\frac{D(P)}{n} \mid -1\right)$ . Then the number of  $n$  such that  $\left(\frac{P^2-4Q_0}{p} \mid -1\right)$  and  $k \equiv 0 \pmod{4}$  is given as follows.

- For  $\left(\frac{Q_0}{p} \mid -1\right)$  as  $\left(\frac{d \cdot \{k, p - \epsilon(p)\}}{\nu_2(d) = \nu_2(p - \epsilon(p))} \mid -1\right)$  when  $\left(\frac{Q_0}{n} \mid -1\right)$
- for  $\left(\frac{Q_0}{p} \mid 1\right)$  as  $\left(\frac{d \cdot \{k, p - \epsilon(p)\}}{\nu_2(d) < \nu_2(p - \epsilon(p))} \mid -1\right)$  when  $\left(\frac{Q_0}{n} \mid -1\right)$  otherwise

*Proof.*  $n$  is a psp  $\left(\frac{D(P)}{n} \mid -1\right)$  □

**Theorem 4** If  $n$  is a spsp  $\left(\frac{D(P)}{n} \mid -1\right)$  and  $\left(\frac{Q_0}{p} \mid -1\right)$ , then the number of  $n$  such that  $\left(\frac{P^2-4Q_0}{n} \mid -1\right)$  and  $\frac{n+1}{2} \equiv 0 \pmod{4}$  is

$$i \qquad \qquad \qquad l \qquad i$$

$$\begin{aligned} &- \text{ when } \left(\frac{Q_0}{p}\right) = -1 \text{ given as } \begin{cases} \left(\frac{(n+1,p+1)}{2}\right) & \text{for } 2 \mid 1 \quad 2 \mid 1 \\ 0 & \text{otherwise} \end{cases} \\ &- \text{ when } \left(\frac{Q_0}{p}\right) = 1 \text{ given as } \begin{cases} \left(\frac{(n+1,p-1)}{2}\right) & \text{for } 2 \mid 1 \quad 2 \nmid -1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

$$\begin{aligned} \text{Proof.} \quad & \begin{matrix} 1 & n \\ n+1 & 0 & -0 \\ 2 & 0 & 2 & 1 \\ n & & & \end{matrix} \alpha \begin{matrix} n \\ \frac{n+1}{2} & 0 & -0 \\ n & & \end{matrix} \alpha \begin{matrix} n \\ n \\ n \end{matrix} \\ & \left(\frac{Q_0}{p}\right) = \begin{matrix} -1 & n & 1 \\ 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 \end{matrix} \quad \begin{matrix} 1 & - & 2 & 1 \\ 2 & 1 & 2 & 1 \\ n+1 & 0 & -0 \end{matrix} \alpha \begin{matrix} 1 & n \\ 2 & 1 & 2 & 1 \\ n & & \end{matrix} \end{aligned}$$

$$\left( \begin{matrix} d \nmid (n+1,p+1) \\ \nu_2(d) = \nu_2(p+1) \\ (d, ord_n(Q_0)) = 2 \end{matrix} \right)$$

$$\begin{aligned} & \begin{matrix} n \\ n & n & n & 0 \\ n & & n & \end{matrix} \begin{matrix} \frac{n+1}{2} & 0 & -0 \\ n & & \end{matrix} \alpha \begin{matrix} 0 \\ n \\ n \end{matrix} \\ & \begin{matrix} n & | & | & 1 & n & | & n & 0 & | & -1 \\ n & & n & \frac{(n+1,p+1)}{2} & & & & & & \end{matrix} \\ & \begin{matrix} n \\ n & & n & \end{matrix} \left(\frac{Q_0}{p}\right) = \begin{matrix} 1 & n & 1 & n \\ 2 & -1 & 1 & n \\ 2 & 1 & 2 & -1 \end{matrix} \quad \begin{matrix} 1 & n \\ 2 & 1 & 2 & -1 \\ 2 & 1 & 2 & -1 \end{matrix} \quad \begin{matrix} n \\ n & & \end{matrix} \end{aligned}$$

$$\left( \begin{matrix} d \nmid (n+1,p-1) \\ (d, ord_n(Q_0)) = 2 \end{matrix} \right) - \left( \begin{matrix} d \nmid \frac{n+1}{2}, p-1 \\ (d, ord_n(Q_0)) = 2 \end{matrix} \right)$$

$$1 \quad -1 \quad -1 \quad -\frac{n+1}{2} \quad -1 \quad 1 \quad \frac{1}{2} \quad 1 \quad -1 \quad \square$$

$$n$$

**Theorem 5** Suppose that a composite number  $n$  fulfills for all prime divisors the following conditions

$$\begin{aligned} &- \text{ if } \left(\frac{Q_0}{p}\right) = 1 \text{ then } \begin{cases} 2 \mid 1 & 2 \nmid -1 \text{ and } 2 \nmid -1 & 2 \nmid -1 \\ -1 & \end{cases} \\ &- \text{ if } \left(\frac{Q_0}{p}\right) = -1 \text{ then } \begin{cases} 2 \mid -1 & 2 \mid 1 & 2 \mid 1 \text{ and } 2 \nmid -1 & 2 \nmid -1 \end{cases} \end{aligned}$$

Let  $\gamma = 1$  respectively 0, according as  $\alpha = -1$  or  $\alpha = 1$  modulo  $2$ . Then the number of parameters  $(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)$  that pass the proposed test of section 4.2 is given as

$$\left( \begin{array}{cccc} 1 & & & \\ & 1 & -1 & - \\ & & & 1 \\ & & & & 1 & -\gamma \end{array} \right) \begin{array}{l} \\ \\ \\ \frac{Q_0}{p} = 1 \end{array} \quad \begin{array}{l} \\ \\ \\ \frac{Q_0}{p} = -1 \end{array}$$

Otherwise  $\hat{\theta}_n$  does not pass the proposed test for any of the parameters  $\theta \in \Theta$ .

### 6.3 Some Special Cases

$\begin{array}{ccccccccccc} & & & & & n & & & n & & \text{spsp} \\ & & & & 0 & & & n & & n & & n \\ & & & n & & n & & & & & n & \\ n & & n & & & & & & & & & \end{array}$

**Lemma 4** *Let  $\left( \begin{smallmatrix} r \\ i=1 \end{smallmatrix} \right)_{i=1}^r$  be a sequence of positive integers. Suppose that for some  $r$  there exists a parameter  $i$  with  $\frac{p_i-1}{2} \nmid p_i$  such that  $\left( \begin{smallmatrix} p_i \\ i \end{smallmatrix} \right)$  is a  $\mathbf{psp}$ . A necessary condition for  $\left( \begin{smallmatrix} p_i \\ i \end{smallmatrix} \right)$  to be a  $\mathbf{Lpsp}$  for  $i \in \mathbb{Z}_n$  and  $\left( \frac{D}{n} \right) = -1$ , is that  $\left( \frac{D}{p_i} \right) = -1$ .*

*Proof.*  $\left(\frac{D}{p_i}\right) \equiv 1 \pmod{p_i} \mid i-1 \pmod{p_i} \equiv 1 \pmod{p_i}$

**Corollary 2** *A necessary condition for a Carmichael number  $n$  to be a Lpssp with  $\gcd(n, p_i) = 1$  and  $\frac{D}{n} \equiv -1 \pmod{p_i}$  is that  $\chi \not\equiv 1 \pmod{p_i}$  and  $\frac{D}{p_i} \equiv -1 \pmod{p_i}$  for all  $i$ .*

n

**Lemma 5** *Let  $\left(\begin{smallmatrix} r \\ i=1 \end{smallmatrix}\right)_i$  be a spsp to a set of bases. Suppose that for every  $i$  there is an  $i - i$  with  $i | p_i - i$  where  $i$  denotes the odd part of  $i - 1$ . Let  $2 -$  such that  $\left(\frac{D}{n}\right) = -1$  and  $-0$ . If  $|$  or  $\left(\frac{D}{p_i}\right) = 1$  for some  $i$  then cannot be a sLpsp.*

n

**Corollary 3** *Suppose that for all primes  $p_i$  dividing  $n$  (at least) one of the following conditions holds:*

$$- \left( \frac{Q_0}{p_i} \right) - 1.$$

Then,  $\mathcal{C}_i$  can pass the proposed test of section 4.2, only if, for all  $j \in \mathcal{I}_i$

$$\left(\frac{0}{i}\right) \left(\frac{1}{i}\right) \equiv -1 \pmod{2} \quad \text{and} \quad \left(\frac{1}{i}\right) \equiv -1 \pmod{2}$$

In this case the number of  $\mathcal{C}_i$  that pass the test equals

$$\left(\frac{1}{p_i}\right) \equiv -\gamma \pmod{n}$$

where  $\gamma$  is defined in Theorem 5.

**Remark 2** (i) If  $n$  has  $k$  prime factors, then with probability  $\frac{1}{2^k}$  the base  $a_0$  is a nonresidue for all the  $\mathcal{I}_i$ . Since the **spsp** test is extremely fast, it can very efficiently be run for variations of  $a_0$ . As a second step, now Corollary 3 asserts that the combination with the Lucas test as described in section 4.2 is highly reliable.

(ii) It is known (cf. [3]) that if a composite integer  $n$  is a **spsp** w.r.t. all possible  $\frac{\phi(n)}{4}$  bases, then  $n$  is either of the form  $1 \cdot 1$ , or  $n$  is a Carmichael number with three factors that are  $1 \cdot 1$ . The first from can easily be checked for compositeness, since  $1 \cdot 1$  implies that  $1$  is a perfect square. For numbers of the second form, no efficient algorithms are known. However, in this case Corollary 3 asserts that the proposed test is very effective. In particular, since  $i-1 \mid -1$  the number of liars  $\mathcal{C}_i$  obviously will be very small.

### 6.4 Some Numerical Examples

1	spsp	n	0	n	1	n	n
		n	spsp	$-10^{13}$	n	1	n
		0	n		n		--
					n	n	
		n		n			

Distribution of all composites $n < 10^{13}$ , $n \equiv -1 \pmod{10}$ that pass the proposed test for $a_0$			
n	n	n	n
1		9	
n		n	
10	9	1	0
	9	1	0 1
	1 0	1	0 1
	0	1	0 09
	0 0		0

l i ill i li y i i i  
**64**  
 illi i **35**  
 l i i y i y y y  
 y i i i  
 i l y l i  
 l l i i y  
**72**  
 ill i il i i i l i  
**7**  
 i l **61**  
 l i i **3**  
 i l ll i i  
 i **1**  
 l i lli ll  
 y i **225**  
 ll i i i l  
 i ll i i  
 l l **10**  
 ll i l lli ll  
 y i **225**

i l i  
 ll i th i l i i i  
 li i ll l i i i li y i i  
 i i l y l y i l i l i  
 i i l 15 **61**  
 i l l i i ly i l i i l  
 i ll i i l l **6**  
 i i d i i li i  
 li i i i l l

# On the Cryptanalysis of Nonlinear Sequences

## [Invited Paper]

Communication Sciences Institute  
University of Southern California  
Los Angeles, CA 90089-2565 U.S.A.  
c/o milly@mizar.usc.edu

**Abstract.** A nonlinear boolean function  $f(x_1, x_2, \dots, x_k)$  of  $k$  binary variables may be used in two basically different ways to generate a nonlinear binary sequence, *internally* or *externally*. Internally,  $f$  may be part of the feedback computation of a nonlinear feedback shift register. Externally,  $f$  may be applied to the output bit stream of another sequence generator (e.g. a linear shift register) to introduce nonlinearity, or greater nonlinearity. A third approach is to use  $f$  to obtain a nonlinear combination of  $k$  linear sequences. The vulnerability of systems using  $f$  in any of these ways to cryptanalysis depends on the multidimensional correlations of  $f$  with the modulo 2 sums of the subsets of its variables. This principle was published by the present author in [1] in 1959, and included as Chapter 8 in his book [2] in 1967. It was subsequently rediscovered and republished in 1988 in [3], on the basis of which it is sometimes known as the Xiao-Massey algorithm. Some practical aspects of the use of this principle in code construction as well as code breaking, and for other types of signal design, are discussed.

## 1 Introduction

h  $2^{2^k}$  boolean functions  $k$   $F_2^k$   
 $F_2$  h  $2^k$  linear homogeneous 2  
h  $k$  h  $2^k$   
linear inhomogeneous h  $2^{2^k} - 2^{k+1}$   
nonlinear boolean functions  $k$   
h binary maximum-length linear shift register sequences  $m$ -se-  
quences PN sequences h  
h h  
h h h  $m$ -  
 $2^n -$  h  $n$   $n$ - h h h  
h  $n$  h  $n$  h  
h h h h  
nonlinearity

2 Nonlinear Shift Register Sequences

h  
h h h

2.1 Nonlinear Feedback Shift Registers

h h k h n h  
h h k h nonlinear shift  
register h h  
2^n h h h  
h h h h  
h 2 h h

2.2 Linear Sequences with External Nonlinear Logic

n- h m-  
2^n - h k k  
h 2^n - h h h  
h k h h  
h n h h h h  
h h h h h n  
h h k n

2.3 Nonlinear Combinations of Several Sequences

h k k  
k h h h h  
P p\_1, p\_2, ..., p\_k h h h  
l.c.m. p\_1, p\_2, ..., p\_k h h  
h m- h h h  
h h h h h  
h h h h k h  
h h different



$$\begin{array}{ccccccc}
 H & & h & & h & & G \\
 h & & h & & & & h \\
 & & & h & & h & h \\
 & & h & & & & h \\
 & & & & & & 2
 \end{array}$$

**Theorem 1** Let  $c_0 = \sum_{all\ x_j} f(x_1, x_2, \dots, x_k)$ , which is the number of 1's in the truth table for  $f(x_1, x_2, \dots, x_k)$ , and let  $T_0 = c_0, 2^k - c_0$ . Then  $c_0$  is an invariant for the orbit of  $f$  under  $H_k$ , and  $T_0$  is an invariant for the orbit of  $f$  under  $G_k$ .

*Proof.*

$$\begin{array}{ccccccc}
 & h & & h & h & f & h \\
 & f & & c_0 & 2^k - c_0 & h & c_0 \\
 T_0 & c_0, 2^k - c_0 & & & G & & H
 \end{array}$$

**Definition 1**  $T_0$  and  $c_0$  are the zero order invariants of  $f$  with respect to  $G$  and  $H$ , respectively.

**Theorem 2** For each  $i, 1 \leq i \leq k$ , let  $R_1^i = d_1^i, 2^k - d_1^i$ , where  $d_1^i = \sum_{all\ x_j} f(x_1, x_2, \dots, x_k) \oplus x_i$ . Then the set of numbers  $R_1^1, R_1^2, \dots, R_1^k$ , when arranged in descending order, forms a collection of  $k$  invariants (the "first-order invariants")  $T_1^1, T_1^2, \dots, T_1^k$  for the orbit of  $f$  under  $G$ . These are also invariants for the orbit of  $f$  under  $H$ . (For notational consistency, we may denote these same invariants by  $c_1^1, c_1^2, \dots, c_1^k$ , when referring to them as the "first-order invariants" with respect to  $H$ .) [The symbol  $\oplus$  denotes modulo 2 addition, but the summation sign denotes ordinary addition.]

*Proof.*

$$\begin{array}{ccccccc}
 h & R_1^i & h & h & x_i & & \\
 d_1^i, 2^k - d_1^i & 2^k - d_1^i, d_1^i & h & h & \{R_1^1, R_1^2, \dots, R_1^k\} & & \\
 & & z & h & \{T_1^i\} & h & \{R_1^i\} \\
 & & & & & h & \{T_1^i\} \\
 f & & f & f & x_1, x_2, \dots, x_k \oplus x_i & h & h \\
 x_i & h & h & \{T_1^i\} & H & & \\
 G & & & & & &
 \end{array}$$

**Definition 2** For every pair  $i, j$  with  $1 \leq i, j \leq k$ , define  $d_2^{ij} = \sum_{x_1, x_2, \dots, x_k} f(x_1, x_2, \dots, x_k) \oplus x_i \oplus x_j$ . The complement of  $d_2^{ij}$  is  $2^k - d_2^{ij}$ . That permutation and complementation of the variables of  $f$  (not necessarily unique) which is consistent with yielding the values  $\{R_1^i\}$  in descending order and which makes the sequence of  $\{d_2^{ij}\}$  numerically greatest (i.e. larger values occur ahead of smaller values, to the extent permitted by consistency with the ordering of  $\{T_1^i\}$ ) yields the second-order invariants  $T_2^{1,2}, T_2^{1,3}, \dots, T_2^{2,3}, \dots, T_2^{k-1,k}$  for the orbit of





h

# Securing Aeronautical Telecommunications

## [Invited Paper]

Simon Blake-Wilson

**Abstract.**

—  
— fl  
—

# Tensor-Based Trapdoors for CVP and Their Application to Public Key Cryptography

## (Extended Abstract)

g -

.. 9

{fischlin,seifert}@informatik.uni-frankfurt.de  
<http://www.mi.informatik.uni-frankfurt.de/>

**Abstract.**

## 1 Introduction

v g m -  
m g m w m g m -  
m g w m m m  
g m g m g v w  
m m g g m m - -  
m g m m m g w  
m g m - - m  
m g m g w m g m  
m w w  
g w v g m  
m g v g v  
g w w g  
w m v g  
- m x m  
m v m  
m w m w  
+ g v g  
w w - - m  
g m g  
w m m

$w$   $v$   $g$   $v$   $g$   $g$   $m$   
 $v$   $w$   $g$   $v$   $m$   
 $v$   $v$   $m$   
 $w$

## 2 Lattices, Reduction, and Closest-Vector-Problem

$w$   $v$   $m$   
 $w$   $v$   $m$

**Definition 1 (Lattice)** Given an ordered set (matrix)  $b_1 \dots b_m$  of linear independent column vectors in  $\mathbb{R}^m$ , the set of all integral linear combinations of the vectors

$$\mathcal{L} = \left\{ \sum_{i=1}^m \alpha_i b_i \mid \alpha_i \in \mathbb{Z} \right\}$$

is called a lattice generated by the base  $b_1 \dots b_m$ . Its dimension is  $m$  and if we call it a full dimensional lattice. The vectors  $b_i$  are called lattice points. A lattice  $\mathcal{L}_{\text{sub}} \subset \mathcal{L}$  with  $\dim \mathcal{L}_{\text{sub}} = m$  is a sublattice of  $\mathcal{L}$ . Sublattices of  $\mathcal{L}$  are called integer lattices.

$v$   $g$   $v$   $w$   $-$   
 $g$   $g$   $m$   $v$   $g$   $g$   
 $m$   $x$   $-$   $\left\{ \begin{matrix} g \\ m \end{matrix} \right\} - \dots \}$   $w$

**Definition 2 (Reciprocal Base and Lattice)** If  $b_1 \dots b_m$  is a base for the lattice  $\mathcal{L}$ , then the  $m \times m$  matrix  $B^*$  with  $B^* = B^{-T}$  is called the reciprocal (or dual) base to  $B$ .  $\mathcal{L}^*$  is called the reciprocal lattice to  $\mathcal{L}$ .

$$w = m / \dots m$$

**Definition 3 (Determinant)** The determinant of a lattice  $\mathcal{L}$  with base vectors  $b_1 \dots b_m$  is the  $m$ -dimensional volume of the fundamental parallelepiped  $\sum_{i=1}^m \alpha_i b_i$  which equals  $\sqrt{\det(B^T B)}$  and in case of a full dimensional lattice  $\det(B^T B) > 0$ .

$v$   $m$   $-$   $w$   $v$   $-$   
 $e$   $-$   $g$   $v$   $b_1$   $b$   
 $v$   $m$   $m$   
 $v$   $g$   $g$   $m$   $-e$   $-$   $v$   
 $0$   
 $b_1$   $b$   $w$   $g$   $w$   $m$   $m$   
 orthogonalization  $b_1$   $b$   $w$   $g$   $w$   $m$   $m$   
 $-b$   $b$   $-b$   $-$   $b_1$   $b_1$

Gram-Schmidt or-

$$\begin{array}{cccccccccccc}
 \mathbf{b} & & \mathbf{b} & - & \frac{-1}{=1} & & \mathbf{b} & & & & -\mathbf{b} & - & & \mathbf{g} & & \text{th} & & \mathbf{v} & - \\
 & & & & & & & \mathbf{w} & \mathbf{w} & & & & & & & & & & \\
 & & & & \mathbf{g} & & & & \mathbf{m} & \text{---} & & \mathbf{g} & & & & & & 0 & \\
 & \mathbf{g} & & & \mathbf{m} & - & \mathbf{m} & & \mathbf{m} & & & \mathbf{b}_1 & & \mathbf{b} & & \mathbf{b}_1 & & \mathbf{b} & - & \top \\
 & & \mathbf{v} & & & & =1 & -\mathbf{b} & - & & \mathbf{g} & & \mathbf{v} & & \mathbf{b}_1 & & \mathbf{b} & & \\
 \mathbf{g} & & & & & & & & & & \mathbf{g} & & & & & & -\mathbf{b}_1 & & \mathbf{b} & - & - \\
 & -\mathbf{b}_1 & & \mathbf{b} & - & & \mathbf{v} & & \mathbf{b} & & & & & & & & & & & & 
 \end{array}$$

**Definition 4 (Successive Minima)** The  $i^{\text{th}}$  successive minima of a lattice is the smallest  $\lambda_i > 0$  such that there are  $i$  linear independent lattice points  $\mathbf{v}_1, \dots, \mathbf{v}_i$  with  $\|\mathbf{v}_i\| \leq \lambda_i$ .

$$\begin{array}{ccccccc}
 \mathbf{w} & & \mathbf{v} & \mathbf{g} & & & \mathbf{v} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\
 & & & \mathbf{m} & & & & & & 
 \end{array}$$

**Proposition 1 (Minkowski 1896)** If  $L$  is a lattice, then  $\lambda_1 \leq \sqrt{\frac{\text{vol}(L)}{\dim L}}$ .

$$\begin{array}{ccccccccccc}
 \mathbf{w} & & & & 0 & & \mathbf{m} & & & & \\
 & \mathbf{v} & \mathbf{m} & \mathbf{m} & & & \mathbf{m} & \mathbf{z} & & 1 & \\
 1 & & & \frac{1}{\dim L} & & & \mathbf{m} & \mathbf{x} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\
 & & & & & & \mathbf{m} & & & \gamma & \\
 & & & & & & & & \mathbf{m} & & \mathbf{m} & \mathbf{m} \\
 \mathbf{w} & & \frac{1}{2} & 1 & & \mathbf{v} & & \mathbf{v} & & \mathbf{m} & \mathbf{w} \\
 & \mathbf{g} & & & & \mathbf{v} & \mathbf{m} & \mathbf{m} & & & 
 \end{array}$$

**Definition 5 (Closest-Vector-Problem CVP)** Given a full dimensional lattice  $L$  and a point  $\mathbf{x} \in \mathbb{R}^n$ , the Closest-Vector-Problem is to find a lattice point  $\mathbf{b} \in L$  with minimal distance  $\|\mathbf{x} - \mathbf{b}\|$ .

$$\begin{array}{ccccccccccc}
 \mathcal{N} & - & & & \mathbf{g} & \mathbf{m} & & & \mathbf{v} & \mathbf{g} & - \\
 & & & & & & & & & \mathbf{w} & \\
 & & \mathbf{x} & \mathbf{m} & & \mathbf{v} & \mathbf{w} & & 2 & & \mathbf{g} \\
 & & \mathbf{g} & \mathbf{m} & & & & & \mathbf{x} & \mathbf{m} & \mathbf{m} \\
 \mathbf{v} & & & & & & & & \mathbf{x} & & 0 & \mathbf{g} & \mathbf{m} \\
 & & & & & \mathbf{x} & & \mathbf{w} & & & & \mathbf{w} & \mathbf{w} \\
 & & & & & & & & & & \mathbf{v} & & 
 \end{array}$$

**Lemma 1 (Nearest Plane)** Suppose  $L$  is a full dimensional lattice given by a base  $\mathbf{b}_1, \dots, \mathbf{b}_n$  with  $\|\mathbf{b}_i\| \leq \lambda_i$ . For  $\mathbf{x} \in \mathbb{R}^n$  with  $\|\mathbf{x}\| \leq \lambda_n$  one can efficiently compute the uniquely determined closest lattice vector.

$$\begin{array}{ccccccc}
 \mathbf{m} & & & & \mathbf{v} & & \\
 \mathbf{g} & & \mathbf{g} & & \mathbf{g} & & \mathbf{m} \\
 \mathbf{g} & \mathbf{z} & - & & & & 4
 \end{array}$$

**Definition 6 (Lattice Reduction)** Given a base  $\mathbf{b}_1, \dots, \mathbf{b}_n$  and  $\lambda_i$  denote  $\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_n)$ . We call the base  $\mathbf{b}_1, \dots, \mathbf{b}_n$  reduced if  $\|\mathbf{b}_i\| \leq \lambda_i$ .

a) —  $\frac{1}{2}$  for — .  
 b)  $-b$  — 1 for

There are two special cases: For  $\mathbf{B}^{-1} \mathbf{B} = \mathbf{I}$  it is called LLL base and for  $\mathbf{B} \mathbf{B}^T = \mathbf{I}$  one calls it HKZ base.  $\mathbf{B}$  is a reciprocal  $\delta$ -reduced base, if the reduction holds for the reverse ordered reciprocal base  $\mathbf{B}^*$  [LLS90].

v m v  
 -Z v -Z v g  
 m m g  
 w g m -  
 g v v x  
 m m .. m  
 - - 0 - 00  
 v z m m m

**Proposition 2** ([LLL82, LLS90]) *Let  $\mathbf{b}_1, \dots, \mathbf{b}_n$  be a base of the lattice  $L$ . For*

a) If  $\mathbf{A}$  is LLL reduced, then  $\|\mathbf{b}\| \leq \frac{1}{(-1)^2} = \frac{1}{(-1)^2}$ .

b) If  $\mathbf{A}$  is reciprocal HKZ reduced, then  $\|\mathbf{b}\| \leq \frac{1}{\gamma} = \frac{1}{\gamma}$ .

$$- \frac{w}{4} \frac{1}{m^4} \frac{\gamma}{\gamma} \frac{m}{4}$$

### 3 Tensor Product of Lattices

g 4 w m m w m w v w m

$\mathcal{L} \otimes \mathcal{L}$  w  $\otimes$  w m

- m x m - m x - m x

0 g w x m g

$\otimes$  -  $\otimes$  v

g  $\otimes$  1 2 1 2 2

1 - 1 w m 2 - 2

1 2 2 - 2

$(-1)^{2+1} (-1)^{2+2} (-1)^{2-2}$

1 1 1

$$\begin{aligned} & \frac{m}{m} \frac{v}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & - 0 \end{aligned}$$

**Proposition 3** Suppose  $\Gamma_1, \Gamma_2$  are two full dimensional lattices. Then we have  $m(\Gamma_1 \otimes \Gamma_2) = m(\Gamma_1) \cdot m(\Gamma_2)$  and  $\dim(\Gamma_1 \otimes \Gamma_2) = \dim \Gamma_1 + \dim \Gamma_2$ .

$$\Gamma_1 \otimes \Gamma_2 = \bigoplus_{j \neq i} \Gamma_j$$

**Proposition 4** Suppose  $\Gamma_1, \Gamma_2$  are two full dimensional lattices. Then we have  $m(\Gamma_1 \otimes \Gamma_2) = m(\Gamma_1) \cdot m(\Gamma_2)$  with equality if  $m(\Gamma_1) = 4$  or  $m(\Gamma_2) = 4$ .

# 4 CVP-Based Public Key Cryptosystems

Frame Work.

$$\begin{aligned} & \frac{m}{m} \frac{v}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{x}{m} \frac{m}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{pub}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{v}{m} \frac{pub}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{g}{m} \frac{pub}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{v}{m} \frac{v}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{g}{m} \frac{pub}{m} \frac{g}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{m}{m} \frac{g}{m} \frac{m}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{y}{m} \frac{pub}{m} \frac{m}{m} \frac{e}{m} \frac{m}{m} \frac{e}{m} \\ & \frac{w}{m} \frac{g}{m} \frac{b}{m} \frac{m}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{m}{m} \frac{g}{m} \frac{m}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \\ & \frac{g}{m} \frac{m}{m} \frac{1}{m} \frac{w}{m} \frac{v}{m} \frac{2}{m} \end{aligned}$$

g m m m

gg g m m m m

v m g g m v -

g m g - m

m g g -

g v x w m

pub -

g v g m - ( 3 g2 ( g

g m m m x

m m m w

m m

w m -R

- - -4 4 - w m g w

- 1 m v e -R - w

Finding the Tensor Decomposition.

w g -

g m g m w m

v m m m m g

m v m

v w m

w m m m

g m m m m

w m m m m

w m m m m

w m v w

m m m 4

m v

g g m m 1 L 2 L

m 1 ⊗ 2 m

g v x w v

( dim 1 ( dim 2 -a -b - ( dim 1 -a -dim 2 2

m v g g w

m g g m th m w

m 2 g g m

2 m v -a -dim 2 2

g g g v g

m v v

g m

( dim 1 -dim 2 ( dim 2

Attacks by Lattice Reduction.

m v w

- m +

$$\begin{aligned}
& \mathbf{v} \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{x} \quad \text{pub} \mathbf{m} \quad \mathbf{e} \quad \mathbf{g} \quad \mathbf{m} \\
& \quad \text{ext} \mathbf{g} \quad - \quad \mathbf{m} \quad \mathbf{x} \\
& \quad \text{ext} \left( \mathbf{x} \right)_0^{\text{pub}} \left( \mathbf{e} \right)_0^{\text{pub}} - \quad \mathbf{m} \quad - \quad +1 \\
& \quad \mathbf{m} \quad -\mathbf{z} \quad \mathbf{v} \quad \text{ext} \mathbf{e}_{\text{ext}} - \mathbf{e} \quad \mathbf{w} \\
& \quad \mathbf{x} \quad 2 \quad \text{ext} \quad 1 \quad \text{pub} \quad \mathbf{v} \mathbf{e}_{\text{ext}} \quad \mathbf{v} \\
& \quad \mathbf{x} \mathbf{m} \quad \mathbf{v} \quad 1 \quad \text{ext} \mathbf{w} \quad 1 \quad \text{pub} \quad -\mathbf{e} - \\
& \quad \mathbf{g} \quad \mathbf{g} \quad - \quad \mathbf{m} \quad \mathbf{m} \mathbf{z} \quad -\mathbf{e} - \frac{1}{2} \quad 1 \\
& \quad \mathbf{v} \quad \mathbf{v} \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{m} \quad - \\
& \mathbf{z} \quad \mathbf{v} \quad - \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{x} \mathbf{m} \\
& \quad \mathbf{v} \quad \frac{n-1}{2} \quad - \\
& \quad 4 \quad \mathbf{m} \quad 4 \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{m} \\
& \quad \mathbf{x} \mathbf{m} \quad \mathbf{v} \quad \mathbf{w} \quad \mathbf{m} \quad 0 \\
& \mathbf{m} \quad \mathbf{v} \quad \mathbf{v} \quad + \quad \mathbf{w} \quad \frac{1}{2} \\
& \quad \mathbf{v} \\
& \quad \mathbf{g} \quad \mathbf{g} \quad \mathbf{g} \quad \mathbf{g} \\
& \quad \mathbf{v} \quad \mathbf{w} \quad \mathbf{e} - - - \quad \mathbf{v} \quad \mathbf{v} \\
& \mathbf{m} \mathbf{m} \quad \mathbf{v} \mathbf{g} \quad \mathbf{m} \quad \mathbf{m} \mathbf{y} \quad \text{pub} \mathbf{m} \mathbf{m} \\
& \mathbf{w} - \quad \mathbf{m} \quad - \quad \mathbf{v} \quad \mathbf{m} \mathbf{w} \quad \mathbf{v} \quad \mathbf{g} \quad \mathbf{w} \\
& \frac{1}{2} \quad \mathbf{m} \quad \mathbf{v} \\
& \quad \mathbf{v} \quad \mathbf{e} - - - \quad \mathbf{x} \mathbf{m} \quad \mathbf{m} \mathbf{e} -_{\mathbf{R}} -
\end{aligned}$$

Comparison with McEliece Scheme.

$$\begin{aligned}
& \mathbf{w} \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{v} \quad \mathbf{g} \\
& - \quad \mathbf{m} \quad \mathbf{w} \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{g} \\
& \quad \mathbf{w} \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{g} \\
& \quad \mathbf{v} \quad \mathbf{g} \quad \mathbf{g} \quad - \quad \mathbf{m} \quad \mathbf{x} \\
& \quad \mathbf{g} \quad - \quad 0 \quad - \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{m} \mathbf{m} \mathbf{g} \\
& \quad \mathbf{m} \quad \mathbf{x} \quad \mathbf{m} \quad \text{pub} \quad \mathbf{v} \\
& \quad \mathbf{w} \quad - \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{x} \\
& \mathbf{m} \quad \mathbf{x} \quad \mathbf{g} \quad \mathbf{g} \mathbf{m} \quad \mathbf{x} \quad \mathbf{m} \quad \mathbf{g} \quad - \\
& \quad \mathbf{w} \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{y} \quad \mathbf{m} \quad \text{pub} \quad \mathbf{r} \quad \mathbf{v} \quad \mathbf{m} \\
& \quad \mathbf{x} \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{w} \\
& \quad \mathbf{m} \quad \mathbf{x} \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{x} \quad \mathbf{g} \quad \mathbf{m} \quad \mathbf{x} \quad \mathbf{v} \\
& \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{v} \quad \mathbf{w} \quad \mathbf{m} \mathbf{m} \mathbf{g} \\
& \mathbf{w} \mathbf{g} \quad \mathbf{w} \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{g} \quad \mathbf{v} \quad \mathbf{g} \\
& \quad \mathbf{m} \quad \mathbf{w} \quad \mathbf{g} \quad \mathbf{v} \quad \mathbf{m} \\
& \quad \mathbf{w} \mathbf{g} \quad \mathbf{m} \quad \mathbf{g} \quad \mathbf{v}
\end{aligned}$$

---


$$\tau \geq \qquad \mathbf{b} \in L \setminus \{ \} \qquad \| \mathbf{b} \| \leq \tau \cdot \lambda_1 L \qquad L$$

$v$   $g$   $w$   $w$   $g$   
 $m$   $v$   $g$   $m$   
 $g$   
 $g$   $m$   $m$   
 $m$   $m$   $g$   $m$   $w$   $m$   $g$   $m$   
 $m$   $w$   $-$   $w$   $m$   $m$   
 $m$   $x$   $v$   $w$   
 $g$   $m$   $m$   $w$   $g$   $m$   $g$   $m$   
 $w$   $m$   $m$   $m$   
 $g$   $v$   $m$   $w$   $m$   
 $g$   $v$   $g$   $m$   $g$

### 5 HKZ-Tensor-Trapdoor

$g$   $g$   $g$   $g$   
 $g$   $v$   $m$   $w$   $m$   
 $-$   $w$   $g$   
 $\otimes$   $w$   $-$   
 $g$   $v$   $x$   $w$   $w$

**Proposition 5** Suppose  $\mathcal{B}$  and  $\mathcal{C}$  are bases of two lattices with  $q$  and  $p$  respectively. For the base  $\mathcal{B}$  of  $\mathcal{L}$   $\otimes \mathcal{L}$  we have  $\mathcal{C}$ .

$g$   $m$   $m$   $g$   
 $w$   $m$   $1$   $w$   $m$   $-4$   $m$   $-$   
 $w$   $m$   $v$   $m$   $g$   $g$   
 $w$   $g$   $w$   $m$   $-$   
 $w$   $m$   $x$   
 $x$   $w$   $m$   $w$   $m$   
 $g$   $1$   
 $1 \otimes 2 \otimes \otimes$   
 $\mathcal{L}$   $w$   
 $v$   $(_{=1}^t ((_{=1} (g 4$

$(_{=1}^1$   $\mathcal{B}$   $(_{=1}^1 \frac{1}{\gamma} \frac{1}{\gamma}$   
 $w$   $g$   $-$   $w$   $m$   $g$   $v$   $-$   
 $g$   $w$   $v$   $w \frac{1}{\gamma^t} 1$   $-$   
 $v$   $g$   $v$   $m$   $g$   $m$   
 $x$   $m$   $v$   $w$   $\gamma$   $0$   $-$   $0$   $g$   
 $g$   $m$   $g$   $w$   $-$   $m$



7 Conclusions and Open Problems

v gg w - - m  
m g m -  
g m v  
v x m m g  
m m v g w -  
m v m g g m -  
m v w 1 2 v g v  
m 1 2 m v 1 ⊗ 2  
w g m w  
m m 0

References

On Lovász' Lattice Reduction and the Nearest Lattice Point Problem 9

9 Cryptanalysis of the Original McEliece Cryptosystem 9 99 99

9 An Introduction to the Geometry of Numbers 99

9 A Course in Computational Algebraic Number Theory 99

Sphere Packings, Lattices and Groups 9

Hermite Normal Form Computation using modulo Determinant Arithmetic 9 9

9 Succinct Certificates for Almost all Subset Sum Problems 9 9

9 Public-Key Cryptosystems from Lattice Reduction Problems 9 9

Minkowski's Convex Body Theorem and Integer Programming 9

9 Arithmetic of Quadratic Forms 99

Factoring Polynomials with Rational Coefficients 9

9 Korkin-Zolotarev Bases and successive Minima of a Lattice and its Reciprocal Lattice 99

9 Handbook of Matrices 99

The Theory of Error Correcting Codes 9

9 Les Réseaux Parfaits des Espaces Euclidiens 99

9 Lattice Points in high-dimensional Spheres 99

99	<i>Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97</i>	99
9	<i>Matrix Groups</i>	9 99
	<i>with cyclic Factor Group</i>	9 9
	<i>On Decoding Iterated Codes</i>	9
	<i>A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms</i>	9
9	<i>Block Reduced Lattice Bases and Successive Minima</i>	99
9	<i>Attacking the Chor-Rivest Cryptosystem by improved Lattice Reduction</i>	9 9
+9	<i>Attacks on the GGH Cryptosystem</i>	9 99
	<i>Encryption by Random Rotations</i>	9 9
9	<i>Cryptography: Theory and Practice</i>	99

## A Quasi-Random Orthogonal Matrix

$x$   $4$   $w$   $m$   $w$   $g$   $m$   $g$   $m$   $-$   
 $g$   $g$   $g$   $m$   $m$   $m$   $m$   
 $g$   $-$   $m$   $g$   $m$   
 $m$   $m$   $x$   $T$   $w$   $-$   $m$   $x$   $w$   $-$   
 $-$   $1$   $g$   $m$   $x$   $g$   
 $g$   $w$   $m$   $g$   $m$   $m$   $x$   
 $v$   $w$   $m$   $m$   $m$   $-$   
 $4$   $-$   $g$   $m$   $x$   
 $m$   $m$   $-$   $g$   $m$   $-w$   $-$   
 $-$   $-$   
 $w$   $m$   $fl$   $g$   $w$   $m$   $m$   
 $m$   $x$   $w$   $g$   $g$   
 $pub$   $-$   $-$   
 $w$   $v$   $\mathcal{L}$   $pub$   $-$   $\mathcal{L}$   $-$   
 $v$   $m$   $m$   $g$   $-$   
 $e$   $w$   $-e$   $-$   $w$   $m$   $-e$   $-$   $m$   $x$   $-$   
 $v$   $m$   $m$   $m$   $m$   $m$   $w$   
 $v$   $g$   $m$   $m$





$$\begin{array}{ccccccc}
 0 & - & m & & 0 & & g \\
 & & & & & & \\
 & & \mathbf{a} \in_{\mathbb{R}} [1 \quad )^n & 1 & \mathbf{a} & 0 & - & \overset{1}{\rule{1cm}{0.4pt}} \\
 w & 1 & 0 & m & & w & & m & 1 \\
 m & & \mathbf{a} & -_{\mathbb{R}} & w & & - & & \\
 & & 0 & - & 1 & \mathbf{a} & - & - & 1+1+\frac{1}{n} \\
 w & & & & v & g & \mathbf{a} & & ( \quad +1) & +1 & - \\
 w & & & & & & & & & & 
 \end{array}$$

# Delegated Decryption

Yi Mu, Vijay Varadharajan, and Khan Quac Nguyen

School of Computing and IT, University of Western Sydney, Nepean,  
P.O.Box 10, Kingswood, NSW 2747, Australia  
`yimu,vijay,qnguyen @cit.nepean.uws.edu.au`

**Abstract.** This paper proposes a new public key based system that enables us to have a single public key with one or more decryption keys and a unique signing key. One straightforward application for our system is in delegated or proxy based decryption. The proxy based decryption requires that the decryption authority can be delegated to another party (proxy) without revealing the signing key information. This suggests that the proxy who has the legitimate right for decryption cannot sign on behalf of the public key owner; only the legitimate signer can be the owner of the public key.

## 1 Introduction

Public key cryptography is generally considered to be asymmetric, because the value of a public key differs from the value of its corresponding secret key. One paramount achievement of public key cryptography lies in the fact that it enables both encryption and signature, where a secret key can be used for both decryption and signing, while its corresponding public key can be used for both encryption of a message and verification of signature. It is clear that in a public key system a party who can decrypt a message must possess the associated secret key that can also be used to sign.

Let's consider the situation where you want some party to check your encrypted email messages, whereas you do not want the party to sign on your behalf. The common solution for this situation is to disable the signing function of the secret key by explicitly specifying the public key as encryption only and the secret key as decryption only. On the other hand, you need to have another secret-public key pair for yourself, which can be used for signing. This approach is not convenient due to the maintenance of two key pairs and the separation of public key's duties. Moreover, for some popular email software such as PGP and PEM you can not disable the signing capacity of a secret key.

It is quite clear that we want a public key system where several secret keys map onto a single public key. This kind of mapping can actually be found in group signatures[1,2,3,4]. In a group signature system, any one in the group can sign using its secret key on behalf of its group. The signatures can be verified with the unique group public key. However, we can not use the method of group signatures in our system, because all existing group signature schemes cannot be used for encryption.

In this paper, we propose a proxy decryption system that addresses the problem raised above. In our system, there are two different types of secret keys associated with a unique public key. They are called *owner secret key* and *proxy secret key(s)* respectively. The owner secret key is to be used for signing, whereas the proxy secret key(s) are to be used for decryption only. Our scheme relies on the difficulty in solving discrete logarithm problems associated with polynomial functions. Both the encryption and signing methods are based on ElGamal algorithms[5].

There are a couple of immediate applications of our scheme. First, let us consider a boss-secretary environment. The boss (say, Bob) in an organisation could be too busy to read his emails and may allow his secretary or proxy (say, Alice) to handle his encrypted incoming emails; However Bob does not allow Alice to sign. Second, consider an electronic commerce environment. In a bank, it may be necessary that majority of staff can receive and verify signed and encrypted electronic cash and cheques, whereas only some of them are allowed to sign/issue electronic cash and cheques on behalf of the bank.

The rest of this paper is organised as follows. Section 2 describes the cryptographic polynomial functions required for our scheme. To simplify our presentation we introduce the concept of vectors. The main task of this section is to prove that the designated vectors used in our system are secure. Section 3 outlines the setup of our system and defines the cryptographic keys to be used. Section 4 is devoted to the proxy decryption scheme and the associated digital signature scheme. Section 5 describes two interesting extensions to the system, where we study how to combine a signature with an encryption. We also study a proxy signature scheme, where a proxy (or proxies) can sign on behalf of the key owner in such a way that the signer can be recognised as the legitimate proxy by the signature verifier. The final section is the conclusion.

## 2 Preliminaries

In this section, we discuss the polynomial functions and introduce the concept of vector space.

### 2.1 Construction of Polynomial Functions

Throughout the paper we use the following notations:  $p$  is a large prime,  $\mathbb{Z}_p^*$  is a multiplicative group of order  $p-1$ , and  $g \in \mathbb{Z}_p^*$  is a generator.

Given vectors  $\mathbf{a} = (a_0 \ a_1 \ \dots \ a_{p-1}) \in (\mathbb{Z}_p)^p$  and  $\mathbf{b} = (b_0 \ b_1 \ \dots \ b_{p-1}) \in (\mathbb{Z}_p)^p$ , where  $a_i, b_i \in \mathbb{Z}_p$ , the inner product of  $\mathbf{a}$  and  $\mathbf{b}$  is defined as  $\mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{p-1} a_i b_i \pmod{p}$ . Vectors  $\mathbf{a}$  and  $\mathbf{b}$  are called orthogonal, iff the inner product  $\mathbf{a} \cdot \mathbf{b} = 0 \pmod{p}$ .

**Lemma 1.** Given  $a_0, a_1, \dots, a_{p-1} \in \mathbb{Z}_p$ , let

$$f(x) = \prod_{i=0}^{p-1} (x - a_i)$$

$$\begin{aligned}
 x_1 &= \sum_{i=1}^n \prod_{j=1}^m (-x_j^{a_{ij}}) \\
 &\vdots \\
 x_{-2} &= \sum_{i=1}^n (-x_i^{a_{-2i}}) \prod_{j=1}^m (-x_j^{a_{-2j}}) \\
 x_{-1} &= \sum_{i=1}^n (-x_i^{a_{-1i}}) \prod_{j=1}^m (-x_j^{a_{-1j}}) \\
 &= 1
 \end{aligned} \tag{1}$$

Then the vector  $\mathbf{a} = (a_{01}, a_{02}, \dots, a_{0n})$  is orthogonal to the vector  $\mathbf{x}^{(0)} = (1, x_1^2, x_2^2, \dots, x_n^2)$ , i.e.  $\mathbf{a} \cdot \mathbf{x}^{(0)} = 0$  for  $x_i = 1$ .

*Proof:* Given  $x_i = 1$ , consider the function  $f(\mathbf{x}) = \prod_{i=1}^n (-x_i^{a_{i0}})$ . We have  $f(\mathbf{x}) = \prod_{i=1}^n (-x_i^{a_{i0}}) = \prod_{i=1}^n (-x_i^{a_{i0}}) + (\sum_{i=1}^n \prod_{j=1}^m (-x_j^{a_{ij}}) + \dots + \sum_{i=1}^n (-x_i^{a_{i0}})^{-1} + \dots + x_0 + x_1 + x_2^2 + \dots + x_n^2)$ . Therefore,  $\sum_{i=0}^n x_i^{a_{i0}} = 0$  or  $\mathbf{a} \cdot \mathbf{x}^{(0)} = 0$ .  $\square$

We can see that, if  $x_1, \dots, x_n$  are given, it could be possible to find  $x_0, \dots, x_n$ , by solving the equations  $\mathbf{a} \cdot \mathbf{x}^{(i)} = 0$ . The question then is how to hide the information of  $\mathbf{a}$  such that all vectors  $\mathbf{x}^{(i)}$  that are orthogonal with  $\mathbf{a}$  become computationally hard to determine. More precisely, we require our system to satisfy the following criteria:

- (a)  $\mathbf{a} = (a_{01}, a_{02}, \dots, a_{0n})$  is hidden such that it is computationally hard to be determined.
- (b) It is computationally hard to determine a vector  $\mathbf{x}^{(i)}$  such that  $\mathbf{a} \cdot \mathbf{x}^{(i)} = 0$ .
- (c) Given some hidden information of  $\mathbf{a} - (-x_{-1})$  where  $\mathbf{x}^{(i)} \cdot \mathbf{a} = 0 \pmod{-1}$ ,  $x_{-1} = 1$ , it is computationally hard to determine a  $x_{+1} - x_{-1}$  that is not equal to  $x_{+1} = 1$ , and  $\mathbf{a} = (a_{01}, a_{02}, \dots, a_{0n}) - (-x_{-1})^{+1}$  such that  $\mathbf{x}^{(+1)} \cdot \mathbf{a} = 0 \pmod{-1}$ ,  $x_{-1} = 1$ .

The criteria (a) and (b) can be realised due to the difficulty in solving discrete log problems:

**Lemma 2.** Let  $\mathbf{g}(\mathbf{a}) = (g_0, g_1, \dots, g_n) = (x_0, x_1, \dots, x_n) \pmod{-1}$ . Given  $\mathbf{g}(\mathbf{a})$  for  $-1 \leq i \leq n$ , it is computationally hard to determine a vector  $\mathbf{x}^{(i)}$  such that  $\mathbf{a} \cdot \mathbf{x}^{(i)} = 0 \pmod{-1}$  or  $\mathbf{a} \cdot \mathbf{x}^{(n)} = 1 \pmod{-1}$ .

*Proof:* Immediate. Based on discrete logarithms, the difficulty is due to finding  $x = \log_g y$  is hard and finding a  $x$  such that  $\prod_{j=1}^n x_j^{a_{ij}} = 1 \pmod{-1}$  is hard.  $\square$

However, Lemma 2 does not cover Criterion (c). According to Lemma 3 below it is easy to find a  $x_{+1} - x_{-1}$  such that the dimension of  $\mathbf{g}$  can be arbitrarily increased without the need knowing  $x_{-1}, \dots, x_n = 0$ .

**Lemma 3.** Assume that  $\mathbf{g} = (g_0, g_1, \dots, g_{n+1})$  is given as in Lemma 2. Given  $g_{n+1} = -1$ , a new vector  $\mathbf{g} = (g_0, g_1, \dots, g_{n+1})$  can be determined without the knowledge of  $g_i, i = 1, \dots, n$ , such that  $\mathbf{a}' \cdot \mathbf{x}^{(n+1)} = 1 \pmod{p}$ , for  $\mathbf{x} = (x_0, x_1, \dots, x_{n+1})$ ,  $x_i = 1$  for  $i = 0, 1, \dots, n$ , where  $\mathbf{a}' = (a'_0, a'_1, \dots, a'_{n+1})$ .

*Proof:* Let  $\mathbf{a} = (a_0, a_1, \dots, a_{n+1})$  be the coefficients of the polynomial function  $f(x) = \prod_{i=1}^n (x - g_i)(x - g_{n+1}) = \mathbf{a} \cdot \mathbf{x}^{(n+1)}$ . Given the information  $(g_0, g_1, \dots, g_n)$ ,  $f(x) = 0$  for  $x = 1$ , can be determined in the following way:

$$\begin{aligned} \bar{g}_k &= \sum_{i_1 = \dots = i_{n+1-k}} g_{i_1} \dots g_{i_{n+1-k}} \\ &= \sum_{i_1 = \dots = i_{n+1-k}, i_j \leq n} g_{i_1} \dots g_{i_{n+1-k}} + \sum_{i_1 = \dots = i_{n-k}, i_j \leq n} g_{i_1} \dots g_{i_{n-k}} \\ &= \sum_{k=1}^{n+1} g_k = 0 \\ &= -1 \pmod{p} \end{aligned}$$

Note that  $\bar{g}_{n+1} = -1$ . It is easy to check that  $\mathbf{a}' \cdot \mathbf{x}_j^{(n+1)} = 1 \pmod{p}$  for all  $j = 0, 1, \dots, n$ .  $\square$

The problem given in Lemma 3 is serious, as we will see later that it is equivalent to allowing any user to legally add proxies to the system. In Corollary 1, we show that this problem cannot be solved by simply adding a secret salt parameter (say,  $s = -1$ ) to  $\mathbf{g}$  in the way of [1] where  $s$  is hidden by  $\mathbf{g}$ .

**Corollary 4.** Assume that  $g_1, g_2, \dots, g_n$  are solutions of  $f(x) = \sum_{i=0}^n g_i x^i = 0$ , where  $g_0$  may be not equal to 1 (due to the salt). If  $g_0, g_1, \dots, g_n$  are known, given  $g_{n+1} = -1$ ,  $\bar{g}_0, \bar{g}_1, \dots, \bar{g}_{n+1}$  can be determined, without the knowledge of  $g_1, g_2, \dots, g_n$ , such that function

$$f(x) = \bar{g}_0 + \bar{g}_1 x + \dots + \bar{g}_{n+1} x^{n+1} = 1$$

for  $\mathbf{x} = (x_0, x_1, \dots, x_{n+1})$ ,  $x_i = 1$  for  $i = 0, 1, \dots, n$ .

The proof is straightforward and is omitted.

From Lemma 3 and Corollary 1, we conclude that one way to avoid someone illegally adding  $g_{n+1}$  to the set  $g_1, g_2, \dots, g_n$  is to hide one of the elements of  $g_1, g_2, \dots, g_n$  which are defined in Eq. (1). To achieve this, the following method is recommended.

For the given  $(g_0, g_1, g_2, \dots, g_n)$ , we define a new vector  $\mathbf{a} = (a_0, a_1, a_2, \dots, a_n)$ , where  $a_1 = g_1, a_2 = g_2, \dots, a_n = g_n$ . Let  $\mathbf{a}' = (a'_0, a'_1, a'_2, \dots, a'_{n-1})$  and  $a'_n = \sum_{i=1}^{n-1} a'_i = -1$ , then  $\mathbf{a}' \cdot \mathbf{x}_j = 1$ , for all  $j = 0, 1, \dots, n-1$ . Since  $a'_i$  are not given in a clear form, i.e. they are hidden in  $\bar{a}'_i = 1 - a'_i$ , the issues raised in Lemma 3 are solved. For the sake of convenience, we rewrite  $\mathbf{a}'$  as  $\hat{\mathbf{g}}$ , then  $\hat{\mathbf{g}} = -\mathbf{a}' = \mathbf{a} - \mathbf{1}$ .

### 3 System Setup

The system of proxy decryption consists of several senders, a boss and several decryption proxies.

- Encryption. Any party can encrypt a message using the unique public key of the boss.
- Decryption. Either the boss or any legitimate proxy can decrypt the message encrypted with the owner or boss public key.
- Signature. Only the boss can sign a message that can be verified by any recipient using the boss' public key.

Let Bob be the boss and Alice be his secretary, a proxy. Bob is the principal who owns the public key and the associated secret or decryption keys. In order to construct the secret-public key pair, Bob needs to choose a vector  $\mathbf{x}^{(i)}$  and the associated secret vector  $\mathbf{a}$  such that  $\mathbf{a} \cdot \mathbf{x}^{(i)} = 0$  for  $i = 0, \dots, n-1$ . The properties of  $\mathbf{a}$  and  $\mathbf{x}^{(i)}$  are as described in the previous section.

To construct his secret-public key pair for a single-proxy system (without losing generality, let's set  $n = 3$ ), Bob chooses three random numbers,  $x_1, x_2, x_3 \in \mathbb{Z}_p$  and computes  $\mathbf{x}^{(0)} = (1, x_1, x_2, x_3)^T$ .  $p$  must be sufficiently large such that finding discrete log  $\log_g = \log_{g^{-1}}$  is hard. Bob then picks a number  $\beta \in \mathbb{Z}_p$  at random and computes its inverse  $\beta^{-1}$  and proxy parameters  $\mathbf{a}^{(i)} = \beta \cdot \mathbf{x}^{(i)}$ , for  $i = 1, 2, 3$ . Consider the set  $\hat{\mathbf{a}} = (\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \mathbf{a}^{(3)})^T$ , for  $i = 0, 1, 2, 3$ . Namely,  $\hat{a}_0 = a_0$ ,  $\hat{a}_1 = a_1$ ,  $\hat{a}_2 = a_2$  and  $\hat{a}_3 = a_3$ . The public key of Bob is then  $\mathbf{pk} = (\mathbf{g}, \mathbf{g}^{\hat{\mathbf{a}}})$ . Bob keeps  $x_1, x_2, x_3$  and all  $\mathbf{a}^{(i)}$ 's and can use either  $x_1$  or  $x_2$  as his secret decryption key and  $x_3$  as his signing key. Bob gives  $x_3$  and  $\mathbf{pk}$  to Alice who uses  $x_3$  as her secret proxy key and  $\mathbf{pk}$  as her proxy parameter defined as follows:

**Definition 5.** A proxy parameter is an indicator that proves to another party the legitimacy of the proxy. Proxy parameters are public. That is,

- they are not secrets to the public,
- they can be distributed along with signatures, and
- unlike public keys, they do not need to be signed by a trusted authority.

Proxy parameters have to be computed by Bob. Each proxy needs to have a proxy parameter to be used in proxy decryption and proxy signing. We will see later that proxy parameters may actually be kept private when only encryption/decryption is involved in actual applications. They become public only for proxy signatures.

## 4 Proxy Decryption

Conceptually, we describe proxy decryption as follows:

**Definition 6.** Given public key  $\mathbf{pk}$  and its corresponding owner decryption key  $x_1$ , owner signing key  $x_2$  and the proxy key  $x_3$ , the ciphertext  $c = (r, s)$  of message  $m$  can be decrypted using either  $x_1$  or  $x_2$ , i.e.  $m = (c, x_1) = (c, x_2)$ . Only  $x_3$  but not  $\mathbf{pk}$  can be used to sign, i.e.  $s = (m, x_3)$  and  $c = (m, s)$ .

This definition has omitted proxy parameters. Proxy parameters are only needed during a decryption process, whereas encryption involves only the public key of Bob. Bob and Alice can keep their proxy parameters secret if they wish.

### 4.1 Encryption Scheme

Consider a simple situation where three players are involved in the protocol, a sender, Bob and Alice. The public key in the protocol is  $pk = (G, \hat{\cdot})$ . Let  $\mathcal{H}$  be a one way hash function and  $m$  be the message to be sent to Bob by the sender. The idea of this protocol is for the sender to encrypt  $m$  using the public key  $pk$  and produce the corresponding ciphertext that can be decrypted by either Bob or his proxy, Alice.

To encrypt a message  $m$ , the sender computes  $h = \mathcal{H}(m)$  and encrypts  $h$  using Bob's public key to obtain the ciphertext  $c = (c_1, c_2)$ , where  $c_1 = (\hat{\cdot}_0, \hat{\cdot}_1) = 1$  and  $c_2 = h \pmod{p}$ .  $c$  is sent to Bob whose computer then automatically forwards the message to Alice.

Since Bob possesses the secret key,  $c_1 = c_1$  or  $c_2$ , and Alice possesses the proxy key  $c_3$ , either Bob or Alice can obtain  $m$  by decrypting the ciphertext  $c$ . The decryption key used to decrypt  $c_2$  can be obtained by using either Bob's secret key or Alice's proxy key:

$$\begin{aligned} c_2 &= \hat{\cdot}_0 \prod_{j=1}^3 \hat{\cdot}_j^{-c_j} \\ &= \prod_{j=0}^3 \hat{\cdot}_j^{-c_j} \\ &= \prod_{j=0}^3 \hat{\cdot}_j^{-c_j} \\ &= \sum_{i=0}^3 \hat{\cdot}_i^{-c_i} \\ &= 1 \pmod{p} \end{aligned}$$

The last equality holds because  $\sum_{j=0}^3 \hat{\cdot}_j^{-c_j} = 0$ . The message can be recovered by computing  $m = h \pmod{p}$ . Once  $m$  is obtained, Bob or Alice can verify the correctness of the encryption by checking whether  $c_1 = (\hat{\cdot}_0, \hat{\cdot}_1) = \mathcal{H}(m) \pmod{p-1}$ .

**Theorem 7.** *Encryption protocol.*

**Completeness.** *For a given ciphertext  $c$ , if the sender follows the correct procedures, both Bob and Alice will receive the same message.*

**Soundness.** (1) *The sender cannot cheat by producing the ciphertext that can only be decrypted by Alice. (2) The recipient can verify the correctness of the encryption process even if he does not have the group public key.*

*Proof:*

**Completeness.** The proof is straightforward. As shown above, any one with a valid  $pk$  can decrypt the ciphertext  $c$  and hence obtain the message  $m$ .

**Soundness.** (1) Notice that  $\mathbf{a}$  is orthogonal to all  $\mathbf{x}$  and based on the discussion given in Section 2,  $\mathbf{a}$  cannot be modified. Therefore if  $\mathbf{x}$  can decrypt the ciphertext, any  $\mathbf{x}$ ,  $\mathbf{x} = \mathbf{a}$ , can also decrypt it. (2) The proof is as follows: Once the

message is decrypted, Bob or Alice can verify the correctness of ciphertext. It is done as follows:

- Compute  $\hat{c} = \mathcal{H}(c) \pmod{p-1}$ .
- Compute  $\hat{c} = (\hat{c})^1$  for  $i = 1 \dots 4$  and  $\hat{c}_0 = (\hat{c}_0)^1 \pmod{p}$ .
- Reconstruct the group public key  $- = (\hat{c})$ ,  $= 0 \dots 1 \dots 4$ .
- Verify the correctness of the newly constructed public key by checking  $\prod_{j=1}^4 \alpha^j x_j = 1$  with his private key  $-$ .

Since  $\hat{c}$  is computed using  $\mathcal{H}(c)$ , all the verifications are successful if and only if the sender follows the correct procedures.  $\square$

## 4.2 Signing Scheme

The signature scheme is based on the ElGamal signature scheme [5]. In our system, Bob can sign a message and his digital signature can be verified using his public key; whereas Alice cannot sign on behalf of her boss.

We assume that the signer is Bob. The system is set up by Bob in the following manner: Bob chooses a large prime  $p$ , selects a primitive element  $\alpha$ , computes his secret or signing key such that  $x = \sum_{i=0}^4 \alpha^i x_i$  where  $x_i = \alpha^{-1}$ , and sets the public key parameters to the tuple  $(p, \alpha)$  with  $\hat{c} = \sum_{i=0}^4 \alpha^i$ . Note that  $\hat{c}$  can be computed by the verifier; so Bob does not need to inform the verifier of  $\hat{c}$ . Actually, the public key is still in the form of  $- = (\hat{c})$ .

To sign a message  $m$ , Bob carries out a regular ElGamal signing operation, i.e. he first picks a random number  $k \in \{1, \dots, p-1\}$  such that any  $k$  is relatively prime to  $p-1$  and then computes  $r = (\alpha^k) \pmod{p}$  and  $s = \alpha^{-1}(m - k) \pmod{p-1}$  to form the digital signature  $- = (r, s)$ , where  $- = \alpha^k$  and  $- = \alpha^{-1}$ . The signature  $-$  is then sent to the verifier. To verify the signed message,  $\hat{c}$  checks  $\hat{c} = \prod_{j=1}^4 \alpha^j x_j$ . If the equality holds, the signature is accepted.

## 5 Extension

In this section, we describe two interesting extensions to our scheme: sign-encryption and proxy signature.

### 5.1 Proxy Decryption with Sign-Encryption

In a sign-encrypting system encryption is combined with a sender's signature, sometimes called signcryption[6]. The advantage of such a scheme lies in the reduction of computational complexity. In other words, it is computationally more efficient than signing and then encrypting a message.

We propose such a sign-encryption by combining ElGamal signature scheme with our encryption scheme. Assume that the secret-public key pair of the sender consists of  $(x, y)$ , where  $x = \alpha^s \pmod{p}$  and  $y = \alpha^x$ .

To send a message  $m$  to Bob, the sender carries out the following steps:

1. Chooses a secret encryption key  $p = q = 257$ , two random numbers  $r_1, r_2 = 256$  that are relatively prime to  $p - 1$ , and  $g = 5$  of order  $p - 1$  a primitive element.
2. Computes:

$$\begin{aligned} c_1 &= g^{r_1} \pmod{p} \\ c_2 &= g^{r_2} \pmod{p} \\ s &= r_1 r_2 \pmod{p} \\ m &= (c_1^{r_1} c_2^{r_2} s^{r_1 r_2}) \pmod{p} \quad \text{where } r = r_1 + r_2 \\ c_2 &= c_2 \pmod{p} \\ &= c_2^{-1} (c_1^{-r_1}) \pmod{p-1} \end{aligned}$$

$r_2$  and  $s$  are known only to the sender. The sign-encrypted message is then  $(c_1, c_2, s)$ .

3. Sends  $(c_1, c_2, s)$  to Bob whose computer may forward it to Alice.

Either Bob or Alice can open and verify the message by computing the decryption key and recovering the message  $m$ :

$$\begin{aligned} &= c_1^{-1} [c_1^{r_1} c_2^{r_2} s^{r_1 r_2}] \\ &= c_1^{-1} \sum_{i=0}^3 c_i^{r_i} \\ &= s^{-1} \\ &= m \pmod{p} \\ &= m \end{aligned}$$

### Theorem 8. Sign-encryption Protocol.

**Completeness.** *If the sender follows the correct process, Bob or Alice will always accept the sign-encrypted message.*

**Soundness.** *A cheating signer cannot convince the verifier to accept the sign-encrypted message. A cheating verifier cannot recover the sign-encrypted message.*

*Proof:*

**Completeness.** The proof is straightforward by inspecting the protocol. Note that the completeness of the protocol follows from the properties of ElGamal algorithm.

**Soundness.** The message recovery is the combination of the ElGamal encryption and signature schemes. We examine the soundness by inspecting both the signature verification part and the encryption part. The encryption part is the ciphertext of  $c_2^{r_2}$  and is based the same scheme used in Protocol 1. According

to the lemmas given in Section 2, only the recipient who holds a valid secret key or proxy key can recover the message. The signature part can be written in a more obvious form:

$$\begin{aligned} \text{Signature: } s_1 &= x_1^{-1} \pmod{p} \\ &= x_1^{-1} (x_2^{-1} - 1) \pmod{p-1} \\ \text{Message recovery: } m &= s_1^{-1} (x_1 x_2) \\ &= x_2 \end{aligned}$$

We find that it is a variant of the ElGamal signature scheme and is equivalent to the signature message recovery [7]. Actually, we can refer to the ElGamal signature as a special form of our scheme by omitting parameters  $x_2^{-1}$  and  $x_2$  from the expressions above. The security in such systems is similar to discrete log problem in the ElGamal signatures. For example, assume that Eve tries to forge a signature for a given message  $m$ , without knowing  $x_1$ . Eve chooses a value  $r$  and then tries to find the corresponding  $s$ ; for this to be successful she must compute the discrete logarithm  $\log_{x_1} r$ .  $\square$

### 5.2 Proxy Signature

In some cases, Bob may allow Alice to sign on his behalf, when say he is not available. This is the so-called proxy signature. Any verifier can use the boss' public key to verify her signature. It is important that proxy signatures should be distinguishable from Bob's signatures.

The concept of proxy signature was first presented in [8]. However, our proxy signature scheme is fundamentally different. We define our proxy signature as follows:

**Definition 9.** *Given public key  $pk_B$  of Bob, proxy secret key  $sk_A$  and the associated proxy parameter  $x_2$  for proxy, the proxy signature on  $m$  is defined as:*  
 $s = (x_1^{-1} - 1) x_2 \pmod{p}$ . *The proxy signature can be verified using  $pk_B$ , i.e.  $m = s^{-1} (x_1 x_2) \pmod{p}$ .*

**The Scheme** There is no additional setup requirement for enabling proxy signatures. Assume that the system is being set up by Bob. The secret key for Bob and Alice are  $x_1$  (or  $x_2$ ) and  $x_3$  respectively. The public key is still  $pk_B = (x_1^{-1} - 1) \pmod{p}$ . To sign a message  $m$ , Alice first picks a set of random numbers  $r_1, r_2, \dots, r_{t-1} = 0 \dots 3$ , such that any  $r_i$  is relatively prime to  $p-1$  and then computes

$$s = x_1^{-1} (x_2^{-1} - 1) \pmod{p} = 1 \cdot 2 \cdot 3$$

and

$$s = x_1^{-1} (x_2^{-1} - 1) \pmod{p-1} = 1 \cdot 2 \cdot 3$$

to form the digital signature  $s = (s_1, s_2)$ . To verify the signed message, the verifier checks

$$\prod_{i=1}^3 x_i^{r_i} \cdot x_4^{r_4} \stackrel{?}{=} x_0^{-1}$$

**Theorem 10.** *Proxy signature protocol.*

**Completeness.** *The receiver of the signature will always receive a correct signature from the corresponding proxy, if the proxy and verifier are honest.*

**Soundness.** *A cheating proxy cannot convince the verifier to accept the signature.*

*Proof:*

Completeness. Immediate.

Soundness. Assume that the proxy signature can be forged by an adversary, Eve. Then given  $m \in \mathcal{M}$ , Eve should be able to find some  $r$  and  $s$  such that

$$\prod_{i=1}^3 \hat{r}_i^{\hat{s}_i} = m \quad \text{where} \quad \hat{r}_i = \frac{r_i}{4} \quad \hat{s}_i = \frac{s_i}{0}$$

Assume that the above equality holds. Then given  $\hat{r}_2, \hat{r}_3, \hat{s}_2$  and  $\hat{s}_3$ , Eve should be able to determine  $\hat{r}_1$  and  $\hat{s}_1$  and the above equation can be rewritten as

$$\hat{r}_1^{\hat{s}_1} \hat{r}_1^{\hat{s}_1} = m \quad \text{or} \quad \hat{r}_1^{\hat{s}_1} \hat{r}_1^{\hat{s}_1} = \frac{m}{\hat{r}_1^{\hat{s}_1}} \quad \text{where} \quad \hat{r}_1 = \frac{r_1}{4} \quad \hat{s}_1 = \frac{s_1}{0}$$

This, however, violates the ElGamal's properties.  $\square$

**Untraceable Proxy Signature** As mentioned in Section 3, each proxy parameter is unique to each proxy. Proxy parameters are not secret, so proxies in the above scheme are traceable. Since proxy parameters are not linked to the identities of proxies, it is easy to make proxies anonymous. In fact, we can slightly modify the protocol so that the proxies also become untraceable:

In the setup step, the proxy also initialises a random number  $\gamma \in \mathbb{Z}_{p-1}$  such that  $\gamma$  has the new form:

$$\gamma = \gamma^{-1} (\gamma^2 - 1) \pmod{p-1} = 1 \ 2 \ 3$$

to form the digital signature  $\sigma = (m, \gamma)$ . To verify the signed message, the following confirmation protocol is needed:

1. The verifier computes  $\hat{r}_i = \prod_{j=1}^3 \hat{r}_j^{\hat{s}_j}$ . Please note that if this step was implemented correctly, we have  $\hat{r}_i = \frac{r_i}{0}^{\gamma}$ . However, because  $\gamma$  is unknown to the verifier, three additional steps are needed to complete the verification.
2. The verifier selects a random number  $\phi \in \mathbb{Z}_{p-1}$  and computes its inverse  $\phi^{-1}$ . The verifier then computes  $\phi = (\frac{r_i}{0}^{\gamma})$  and sends  $\phi$  to the signer.
3. The signer computes  $\hat{r}_i = \phi^{\gamma^{-1}}$  and then sends  $\hat{r}_i$  to the verifier.
4. The verifier checks  $\hat{r}_i \stackrel{?}{=} \frac{r_i}{0}$ .

Obviously, the proxy's signature is untraceable, since verifier cannot link proxy parameters in different signatures to a particular proxy. Nevertheless, the untraceability applies only to users excluding Bob. This is because Bob has full information about  $\gamma$ , and therefore he can obtain  $\gamma$  very easily by checking

$$\hat{r}_i \stackrel{?}{=} \frac{r_i}{0}^{\gamma}$$

**Theorem 11.** *The untraceable protocol:*

**Completeness.** *If the prover and verifier follow the protocol then the verifier accepts the signature as a valid signature of .*

**Soundness.** *A cheating prover cannot convince the verifier to accept the signature.*

*Proof:*

**Completeness.** There are two facts: First, the secret number  $x$  and its inverse  $x^{-1}$  are known to the verifier only. To remove  $x$  in  $\phi$ , the prover or signer is faced with a discrete log problem. Second, only the prover knows the secret number  $\gamma$  and its inverse  $\gamma^{-1}$  and hence can remove  $\gamma$  in  $\phi$ . Therefore, the verifier accepts the signature if  $x^{-1} = \bar{x}$ .

**Soundness.** Assume that a cheating signer who computes  $\phi$  without using one of proxy or owner secret keys and thus generates  $\phi = ( \dots \gamma )$ . This will result in a value other than  $\bar{x}^\gamma$  in the verification:  $\prod_{i=1}^3 \hat{e}( \bar{x}_i, \bar{x}_i^{-1} )^{\gamma_j} \stackrel{\text{def}}{=} \dots$ . The verifier then selects  $( \bar{x}^{-1} )$  and computes  $\phi \pmod{\bar{x}}$  that is then sent to the cheating signer. One way that he can convince the verifier is to find  $\phi$  and compute  $( \bar{x}^{-1} )$ . However, he needs to solve the discrete log problem,  $\log_{\bar{x}} \phi$ , which is computationally hard to him.  $\square$

It is noted that the proof of knowledge on  $\gamma$  by the signer is equivalent to proving her ability of removing  $\gamma$  from  $\phi$ . Therefore, we can make the verification process non-interactive by adapting non-interactive equality proof of discrete log proposed in [3]. In our case, the prover is the signer, who should prove that she knows  $\gamma$  without revealing  $\gamma$  to the verifier. The common knowledge is the primitive  $x = g$ . The prover will prove that s/he knows the secret  $\gamma$  from  $\gamma \pmod{\bar{x}}$  without revealing  $\gamma$ .

The prover:

- Chooses  $r = r_{-1}$  at random and computes  $\phi = ( \dots ) \pmod{\bar{x}}$ .
- Computes  $\phi = \mathcal{H}( \phi - r ) \pmod{\bar{x} - 1}$  and  $\phi = \phi - \gamma \pmod{\bar{x} - 1}$ .
- Sends  $( \phi )$  with other signature data to the verifier who can verify the knowledge of equality proof, by checking  $\phi = \phi \pmod{\bar{x}}$ , where  $\phi = \prod_{i=1}^3 \hat{e}( \bar{x}_i, \bar{x}_i^{-1} )^{\gamma_j}$ .

Readers are referred to Ref. [3] for details of non-interactive discrete log proof. Please note that the security assurance relies on the fact  $\phi$  can only be computed after  $\phi$  has been computed or after  $\phi$  has been chosen.

## 6 Conclusion

We have proposed a public key based system, where the tasks of decryption and signature have been separated by introducing proxy keys. Our system is based on a special polynomial function whose security properties have been investigated and found to be suitable for our system. Furthermore, we have also extended our scheme to sign-encryption as well as proxy signatures that allow proxies to sign

on behalf of the owner with the condition that their signature can be identified only by their owner. We believe that our system has its potential applicability in electronic commerce.

## References

1. D. Chaum and E. van Heijst, "Group signatures," in *Advances in Cryptology — EUROCRYPT '91*, (New York), pp. 257–265, Springer-Verlag, 1991.
2. L. Chen and T. P. Pedersen, "New group signature schemes," in *Advances in cryptology - EUROCRYPT'94, Lecture Notes in Computer Science 950*, pp. 171–181, Springer-Verlag, Berlin, 1994.
3. J. Camenisch, "Efficient and generalized group signatures," in *Advances in cryptology - EUROCRYPT'97, Lecture Notes in Computer Science 1233*, pp. 465–479, Springer-Verlag, Berlin, 1997.
4. J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology, Proc. CRYPTO 97*, LNCS 1296, pp. 410–424, Springer-Verlag, Berlin, 1997.
5. T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology, Proc. CRYPTO 84*, LNCS 196, pp. 10–18, Springer-Verlag, Berlin, 1985.
6. Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) – cost(signature) + cost(encryption)," in *Advances in Cryptology – CRYPTO '97 Proceedings*, Springer Verlag, 1997.
7. K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem," in *Advances in Cryptology, Proc. EUROCRYPT 94*, LNCS 950, (Berlin), pp. 182–193, Springer-Verlag, 1994.
8. M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proc. of the Third ACM Conf. on Computer and Communications Security*, pp. 48–57, 1996.

# Fast and Space-Efficient Adaptive Arithmetic Coding<sup>\*</sup>

Boris Ryabko<sup>1</sup> and Andrei Fionov<sup>1</sup>

Siberian State University of Telecommunications and Information Sciences  
Kirov St. 86, Novosibirsk 630102 Russia  
{ryabko, fionov}@neic.nsk.su

**Abstract.** We consider the problem of constructing an adaptive arithmetic code in the case when the source alphabet is large. A method is suggested whose coding time is less in order of magnitude than that for known methods. We also suggest an implementation of the method by using a data structure called “imaginary sliding window”, which allows to significantly reduce the memory size of the encoder and decoder.

## 1 Introduction

Arithmetic coding is now one of the most popular methods of source coding. Its basic idea was formulated by Elias in the early 1960s (see [1]). However, the first step toward practical implementation was made by Rissanen [2] and Pasco [3] in 1976. Soon after that in [4,5,6] the modern concept of arithmetic coding was suggested. The method was further developed in a number of works, see, e.g., [7,8].

The advantage of arithmetic coding over other coding techniques is that it allows to attain arbitrarily small coding redundancy per one source symbol at less computational effort than any other method (the redundancy is defined as the difference between the mean codeword length and the source entropy). Furthermore, arithmetic coding may easily be applied to sources with unknown statistics, when being combined with adaptive models in an adaptive coding scheme (see, e.g., [9,10,11,12]).

The main problem in linking arithmetic coding with adaptive models is the following. All known adaptive models provide some estimate of the actual probability  $p(a_i)$  for each letter  $a_i$  over a source alphabet  $A$ . But arithmetic coding is based on cumulative probabilities  $q(a_i) = \sum_{j < i} p(a_j)$ . Since probability estimates change after each coded symbol, cumulative probabilities must always be re-computed. This requires about  $|A|/2$  operations per symbol, where  $|A|$  denotes the size of the alphabet, which affects the speed of coding. The reduction of the coding speed becomes the more noticeable as the alphabet size increases. From this point of view, the alphabet of  $2^8 = 256$  symbols, which is commonly used in

---

<sup>\*</sup> Supported by the Russian Foundation of Basic Research under the Grant no. 99-01-00586

compressing computer files, may already be treated as a large one. With adoption of the Unicode the alphabet size will grow up to  $2^{16} = 65536$  and become sufficiently large to prevent any direct computation of cumulative probabilities. In this paper, we suggest an algorithm that computes cumulative probabilities in  $O(\log |A|)$  time, which is exponentially less than for known methods.

To obtain a specified (arbitrarily small) model redundancy, denoted henceforth by  $r$ , we use an adaptive scheme with sliding window. The sliding window scheme possesses many useful properties and the only disadvantage, that has so far prevented its wide usage, is the necessity to store the entire window in the encoder (decoder) memory. In obtaining arbitrarily small redundancy, the memory allocated to the window becomes a dominating part in space complexity of the encoder and decoder, which thus grows as  $O(1/r)$ . To remedy this disadvantage we use a type of the scheme called “imaginary sliding window” (the concept of which was presented in [13]). This approach allows to preserve all the properties of sliding window without storing the window itself.

The space complexity ( $S$ ) and time complexity ( $T$ ) of the proposed method in case of a memoryless source, seen as functions of two variables, the alphabet size  $|A|$  and redundancy  $r$ , are upper bounded by the following estimates:

$$S < \text{const} \left( |A| \log \frac{|A|}{r} \right) \text{ bits},$$

$$T < \text{const} \left( \log \frac{|A|}{r} \left( \log |A| + \log \log \frac{|A|}{r} \log \log \log \frac{|A|}{r} \right) \right) \text{ bit operations}$$

(here and below  $\log x = \log_2 x$  and  $\text{const}$  denotes some (different) constants greater than 1). Generalizations toward Markov or tree sources are straightforward and can be done by known techniques.

The paper is organized as follows. In Sect. 2 we consider the problem of combining a conventional sliding window scheme with arithmetic coding. We describe a method for fast operation with cumulative probabilities and investigate its complexity. In Sect. 3 we present an imaginary sliding window scheme which dispenses with the need for storing the window.

## 2 Fast Coding Using Sliding Window

Let there be given a memoryless source generating letters over the alphabet  $A = \{a_1, a_2, \dots, a_n\}$ —with unknown probabilities  $p(a_1), p(a_2), \dots, p(a_n)$ . Let the size of the alphabet be a power of two,  $n = 2^m$  (it is not a restrictive assumption, because, if  $n < 2^m$ , the alphabet may be expanded with  $2^m - n$  dummy symbols having zero probability with trivial modifications of the algorithms). Let the source generate a message  $x_1 \dots x_{l-1} x_l \dots, x_i \in A$  for all  $i$ . The window is defined as a sequence of the last  $w$  symbols generated by the source, where  $w$  denotes the size of the window. At the moment  $l$  the window contains the symbols  $x_{l-w} \dots x_{l-2} x_{l-1}$ . During encoding (or decoding), the window “slides” along the message: as a novel symbol is introduced in the window, the oldest one

is removed. Each letter  $a_i - A$  is assigned a counter  $c_i$  of size  $t = -\log(w + n) -$  bits that contains a number of occurrences of  $a_i$  in the current window plus 1 (to prevent zero probability). In these settings the sum of all counters is equal to the window size plus the alphabet size,  $\sum_{i=1}^n c_i = w + n$  and the estimates of the symbol probabilities  $\hat{p}(a_1), \hat{p}(a_2), \dots, \hat{p}(a_n)$  may be obtained as  $\hat{p}(a_i) = c_i/(w + n)$  for all  $i$ .

Denote the novel symbol to be encoded by  $u$  and the oldest symbol stored in the window, to be removed, by  $v$  (at the moment  $l$ ,  $u = x_l$  and  $v = x_{l-w}$ ). The adaptive encoding of the symbol is performed as follows: encode  $u$  according to the current estimated probability distribution ( $\hat{p}(u) = c(u)/(w + n)$ ); decrement counter  $c(v)$  corresponding to the letter  $v$ ; remove  $v$  out of the window; increment counter  $c(u)$  corresponding to the letter  $u$ ; introduce  $u$  in the window. The decoder, provided that it starts with the same counter contents as the encoder, decodes the symbol according to the current probability estimates and updates the counters in the same way as the encoder.

Encoding of a symbol given an estimated probability distribution may efficiently be carried out by means of arithmetic coding (see [7,8] for more details). This technique, however, requires that cumulative range  $[Q_i, Q_{i+1})$  be specified for the symbol  $u = a_i$ , where  $Q_i$  are defined as follows:

$$Q_1 = 0, \quad Q_i = \sum_{j < i} c_j, \quad i = 2, 3, \dots, n + 1. \quad (1)$$

The direct calculation of  $Q_i$  using (1) requires  $O(tn)$  bit operations. Below we describe a method that allows to reduce complexity down to  $O(t \log n)$ .

Let us store not only the counters (denote this vector by  $\mathbf{C}^1$ ), but also the sums of successive pairs of counters ( $\mathbf{C}^2$ ), the sums of successive quadruples ( $\mathbf{C}^3$ ), and so on. For example, if  $n = 8$ , we need to store the following vectors

$$\begin{aligned} \mathbf{C}^1 &= (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8), \\ \mathbf{C}^2 &= ((c_1 + c_2), (c_3 + c_4), (c_5 + c_6), (c_7 + c_8)), \\ \mathbf{C}^3 &= ((c_1 + c_2 + c_3 + c_4), (c_5 + c_6 + c_7 + c_8)). \end{aligned}$$

By using  $\mathbf{C}$  the values of, e.g.,  $Q_4$  and  $Q_8$  can be computed as

$$\begin{aligned} Q_4 &= c_1 + c_2 + c_3 = C_1^2 + C_3^1, \\ Q_8 &= c_1 + c_2 + \dots + c_7 = C_1^3 + C_3^2 + C_7^1. \end{aligned}$$

It is easily seen that computation of any  $Q_i$ ,  $i = 2, 3, \dots, 8$ , requires to sum up at most  $3 = \log 8$   $t$ -bit numbers.

In general case, we store vectors  $\mathbf{C}^1, \mathbf{C}^2, \dots, \mathbf{C}^m$ ,  $m = \log n$ , which requires  $(2n - 2)t$  bits of memory. Let us show how to compute  $Q_{k+1}$ ,  $k = 1, 2, \dots, n$  (recall that  $Q_1 = 0$ ). Let  $k$  have a binary expansion  $k_m k_{m-1} \dots k_1$ , where each  $k_i = 0$  or 1. Then

$$Q_{k+1} = k_m C_{2k/n}^m + k_{m-1} C_{4k/n}^{m-1} + k_{m-2} C_{8k/n}^{m-2} + \dots + k_1 C_k^1$$

(note that only those  $C_j^i$  for which  $k_i = 1$  are included in the computation). This sum has at most  $m = \log n$   $t$ -bit addends, therefore the complexity of computing cumulative counts is  $O(t \log n)$  bit operations.

When incrementing or decrementing counters in the adaptive scheme, all the elements of the vectors which depend on those counters should also be incremented or decremented. More exactly, incrementing (decrementing)  $c_k = C_k^1$  must be accompanied by incrementing (decrementing)  $C_{k/2}^2, C_{k/4}^3, \dots, C_{k/(n/2)}^m$ , which requires also  $O(t \log n)$  bit operations. To increment, e.g.,  $c_3$  ( $n = 8$ ) we must increment not only  $C_3^1$ , but also  $C_2^2$  and  $C_1^3$ .

Finally, the last operation which relies on cumulative counts is finding an interval  $I(z) = [Q_i, Q_{i+1})$  that contains a given number  $z$  (and the letter  $u = a_i$  to whom this interval corresponds). This problem arises in arithmetic decoding. Let us show how one can solve this problem using the vectors  $\mathbf{C}$  in  $O(t \log n)$  time. First compare  $z$  with  $C_1^m$ . If  $z < C_1^m$  then the target interval lies within  $[0, C_1^m)$ , otherwise it belongs to  $[C_1^m, (w+n))$ . Go on comparing  $z$  with  $C_1^{m-1}$  in the former case, and with  $(C_1^m + C_3^{m-1})$  in the latter, which determines one of the intervals  $[0, C_1^{m-1})$ ,  $[C_1^{m-1}, C_1^m)$ ,  $[C_1^m, C_1^m + C_3^{m-1})$ , or  $[C_1^m + C_3^{m-1}, (w+n))$ . After  $m = \log n$  steps we obtain an interval corresponding to one letter.

The following theorem establishes the computational complexity of the proposed adaptive scheme as function of two arguments, the alphabet size  $n$  and redundancy  $r$ .

**Theorem 1** *Let there be given a memoryless source generating letters over an alphabet of size  $n$ . Let the adaptive scheme based on sliding window and arithmetic coding be applied to the source providing a specified redundancy  $r$ . Then the memory size of the encoder (decoder)  $S$  and encoding (decoding) time  $T$  are given by the estimates*

$$S < \text{const} \left( \frac{n}{r} \log n + n \log \frac{n}{r} \right), \quad (2)$$

$$T < \text{const} \left( \log \frac{n}{r} \left( \log n + \log \log \frac{n}{r} \log \log \log \frac{n}{r} \right) \right). \quad (3)$$

*Proof.* The redundancy  $r$  consists of two parts: the one caused by replacing unknown symbol probabilities by their estimates, denote it by  $r_m$  (model redundancy), and the other caused by the encoder, denote it by  $r_c$  (coding redundancy),  $r = r_m + r_c$ .

It is known (see, e.g., [14]) that for the sliding window scheme  $r_m < \text{const} n/w$ . For arithmetic coding  $r_c < \text{const} n \tau 2^{-\tau}$  [8], where  $\tau$  is an internal register size which must be  $O(t)$ , say,  $\tau = t + \log t + \text{const}$ . For  $t$  we have  $\log(w+n) - t < \log(w+n) + 1$ . So we easily obtain that  $r_c < \text{const} n/w$ . Hence,  $r < \text{const} n/w$  and  $w < \text{const} n/r$ .

The memory size of the encoder and decoder is caused by the necessity to store the window, which requires  $w \log n < \text{const}(n/r) \log n$  bits, the vectors  $\mathbf{C}$ , which requires  $(2n-2)t < \text{const} n \log(w+n) < \text{const} n \log(n/r)$  bits, and internal registers for arithmetic coding, which requires  $O(\tau) < \text{const} t < \text{const} \log(n/r)$  bits. Summing up dominating complexities gives (2).

The coding time per symbol is determined by the computations over the vectors  $\mathbf{C}$ , which requires  $O(t \log n) < \text{const} \log(n/r) \log n$  bit operations, and the computation of ranges in arithmetic coding, which uses  $\tau$ -bit multiplication and division, which, in turn, requires  $O(\tau \log \tau \log \log \tau)$  bit operations [8]. Taking into account  $\tau < \text{const} \log(n/r)$  and summing up gives (3). —

**Corollary 1** *If  $n$  is increasing and  $r$  is to remain constant then*

$$\begin{aligned} S &= O(n \log n) \ , \\ T &= O(\log^2 n) \ . \end{aligned}$$

The time estimate is better than that for known methods (where  $T = O(n \log n)$ ).

**Corollary 2** *If  $n$  is fixed and  $r \rightarrow 0$  then*

$$\begin{aligned} S &= O(1/r) \ , \\ T &= O(\log(1/r) \log \log(1/r) \log \log \log(1/r)) \ . \end{aligned}$$

In the next section we show how to improve the memory size estimate to  $O(\log(1/r))$  implementing the concept of imaginary sliding window.

### 3 Imaginary Sliding Window

Let there still be given a memoryless source defined in Sect. 2. We also use the same notions of the window, counters and probability estimation. Denote, additionally, by  $\nu_i$  the number of occurrences of the letter  $a_i - A$  in the current window,  $\nu_i = c_i - 1$  for all  $i$ . Define  $\xi$  to be a random variable taking on values  $1, 2, \dots, n$  with probabilities

$$\Pr\{-\xi = i - = \frac{\nu_i}{w}, \quad i = 1, 2, \dots, n \quad (4)$$

(the problem of generating such a variable will be considered further below).

The encoding of the novel symbol  $u$  is performed as follows: encode  $u$  according to the current estimated probability distribution ( $\hat{p}(u) = c(u)/(w+n)$ ); generate a random variable  $\xi$ ; decrement counter  $c_\xi$ , i.e., the counter corresponding to the  $\xi$ th letter of the alphabet  $A$ ; increment counter  $c(u)$  corresponding to the letter  $u$ .

A distinctive feature of this scheme is that a *randomly chosen* counter is decremented rather than a counter corresponding to the oldest symbol in the window. In this scheme, the context of the window is not used, therefore storing the window is not needed, which saves  $w \log n$  bits of memory. But a question arises whether this scheme is able to represent the source statistics with the precision sufficient to guarantee a specified redundancy? So the next our task is to show that it is, more exactly, we show that the estimated distribution provided by imaginary sliding window converges with exponential speed to the distribution provided by real sliding window.

Denote by  $\Lambda$  the set of all vectors  $\lambda = (\lambda_1, \dots, \lambda_n)$  such that all  $\lambda_i$  are non-negative integers and  $\sum_{i=1}^n \lambda_i = w$ . It is clear that in case of real sliding window  $\nu = (\nu_1, \dots, \nu_n)$  is a random vector that obeys the multinomial distribution

$$\Pr \nu_1 = \lambda_1, \nu_2 = \lambda_2, \dots, \nu_n = \lambda_n = \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i}, \quad (5)$$

where

$$\binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} = \frac{w!}{\lambda_1! \lambda_2! \dots \lambda_n!}$$

(see, e.g., [15]).

In case of imaginary sliding window,  $\nu = (c_1 - 1, \dots, c_n - 1)$  is also a random vector whose distribution is a question to be answered. We shall indicate by superscript  $l$  the state of vector  $\nu$  after the  $l$ th encoded symbol, i.e., in the process of coding vector  $\nu$  assumes the states  $\nu^1, \nu^2, \dots, \nu^l, \dots$ . The next theorem decides on the distribution for  $\nu$  and shows that imaginary sliding window is a sufficiently precise model of real sliding window.

**Theorem 2** *Let there be given a memoryless source generating letters over the alphabet  $A = -a_1, \dots, a_n$  with probabilities  $p(a_1), \dots, p(a_n)$ ; the imaginary sliding window scheme is used with the window size  $w$ . Then*

$$\lim_l \Pr \nu_1^l = \lambda_1, \nu_2^l = \lambda_2, \dots, \nu_n^l = \lambda_n = \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i}, \quad (6)$$

the limit in (6) exists for any initial distribution  $\nu^0 = (\nu_1^0, \dots, \nu_n^0)$ , and there exists a constant  $C < 1$ , independent of  $(\lambda_1, \dots, \lambda_n)$  and  $(\nu_1^0, \dots, \nu_n^0)$ , such that

$$\left| \Pr \nu_1^l = \lambda_1, \nu_2^l = \lambda_2, \dots, \nu_n^l = \lambda_n - \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i} \right| < C^l. \quad (7)$$

*Proof.* Define a Markov chain  $M$  with  $\Lambda$  states, each state being correspondent to a vector from  $\Lambda$  (informally, each vector  $\nu = (\nu_1, \dots, \nu_n)$  of imaginary sliding window has a corresponding state in  $M$ ). A transition matrix for  $M$  is defined as follows:

$$P_{\lambda^1 \lambda^2} = \begin{cases} \frac{\lambda_j^1}{w} p(a_i), & \text{if } \lambda_i^2 = \lambda_i^1 + 1, \lambda_j^2 = \lambda_j^1 - 1, i = j, \\ & \lambda_k^2 = \lambda_k^1, k = i, k = j; \\ \sum_{k=1}^n \frac{\lambda_k^1}{w} p(a_k), & \text{if } \lambda^1 = \lambda^2; \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

This probability matrix corresponds to transition probabilities of imaginary sliding window. The first line in (8) corresponds to the case when the  $j$ th coordinate of the vector  $\nu^l$  is decremented by 1, and the  $i$ th coordinate is incremented by

1, which means introducing the letter  $a_i$  in the window, which, in turn, occurs with probability  $p(a_i)$ . The second line in (8) corresponds to the case when the vector  $\nu^l$  remains intact. This happens if a coordinate  $\nu_k^l$  is first decremented (with probability  $\nu_k^l/w$ ) and then incremented (with probability  $p(a_k)$ ).

The chain  $M$  has a finite number of states and is plainly seen to be non-periodical (see, e.g., [15]). Consequently, there exist limiting probabilities for  $M$ , i.e., such  $\pi_{\lambda}$  that for any  $\mu, \lambda - \Lambda$  there exists the limit

$$\lim_l P_{\mu \rightarrow \lambda}(l) = \pi_{\lambda} \quad (9)$$

independent of  $\mu$  (here  $P_{\mu \rightarrow \lambda}(l)$  denotes the probability of transition from  $\mu$  to  $\lambda$  in  $l$  steps,  $l \geq 1$ ).

It is known (see [15]) that limiting probabilities satisfy the system of equations

$$\begin{cases} \pi_{\lambda} = \sum_{\mu \rightarrow \lambda} \pi_{\mu} P_{\mu \rightarrow \lambda}, & \lambda \in \Lambda, \\ \sum_{\lambda \in \Lambda} \pi_{\lambda} = 1. \end{cases} \quad (10)$$

Next we show that for any  $\lambda = (\lambda_1, \dots, \lambda_n) \in \Lambda$

$$\pi_{\lambda} = \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i}. \quad (11)$$

To do this, put (11) and (8) in (10). As a result,

$$\begin{aligned} & \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i} \\ &= \sum_{i=1}^n \sum_{j=1, j \neq i}^n \binom{w}{\lambda_1, \dots, \lambda_i - 1, \dots, \lambda_j + 1, \dots, \lambda_n} \\ & \quad - \left( \prod_{i=1}^n (p(a_i))^{\lambda_i} \right) \frac{p(a_j)}{p(a_i)} \left( \frac{\lambda_j + 1}{w} p(a_i) \right) \\ & \quad + \sum_{k=1}^n \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \prod_{i=1}^n (p(a_i))^{\lambda_i} \left( \frac{\lambda_k}{w} p(a_k) \right). \end{aligned} \quad (12)$$

Put the equality

$$\binom{w}{\lambda_1, \dots, \lambda_i - 1, \dots, \lambda_j + 1, \dots, \lambda_n} = \binom{w}{\lambda_1, \lambda_2, \dots, \lambda_n} \frac{\lambda_i}{\lambda_j + 1}$$

in (12) and after simple transpositions obtain

$$\sum_{i=1}^n \sum_{j=1, j \neq i}^n \left( \frac{\lambda_i}{\lambda_j + 1} \frac{p(a_j)}{p(a_i)} \right) \left( \frac{\lambda_j + 1}{w} p(a_i) \right) + \sum_{k=1}^n \frac{\lambda_k}{w} p(a_k) = 1.$$

Consequently, (11) is the solution of (10). Taking into account the definition (9) obtain (6).

The second statement of the theorem may be easily obtained from the general proposition on exponential speed convergence to a limiting distribution (see [15]) (this requires that a Markov chain should satisfy certain conditions which is easily checked in our case). —

From the first proposition of the theorem it follows that imaginary sliding window behaves asymptotically just like as real sliding window or, more exactly, the distribution of  $(\nu_1^l, \dots, \nu_n^l)$  approaches, as  $l$  increases, the distribution (5) of the numbers of occurrences of the letters  $a_1, \dots, a_n$  in the real window. From the second proposition it follows that in case of change in the source statistics at the moment  $l$ , the distribution of  $(\nu_1^{l+\delta}, \dots, \nu_n^{l+\delta})$  converges to the new distribution as  $\delta$  increases ( $(\nu_1^l, \dots, \nu_n^l)$  being thus considered as an initial distribution). Therefore, applied to memoryless sources, imaginary sliding window has the same properties as real sliding window, namely, it allows to estimate the source statistics quite precisely and adapts fast if the statistics change.

Let us now consider the problem of generating random variable  $\xi$ . Assume that the only source of randomness is a symmetric binary source generating a sequence  $z_1 z_2 \dots z_k \dots$ , where each  $z_k \in \{-0, 1-\}$ ,  $\Pr -z_k = 0 = \Pr -z_k = 1 - = 1/2$ , and all symbols are independent. We do not consider here the complexity of obtaining random bits. Our aim is to convert a random bit sequence into a random variable  $\xi$  with probability distribution (4). Denote by  $z$  a random number built from  $s = -\log w$ -random bits, i.e., the binary expansion of  $z$  is  $z_1 z_2 \dots z_s$ . It is plain that  $z$  may take on any value from  $-0, 1, \dots, 2^s - 1$ -with probability  $1/2^s$ . It is also plain that, for any  $w < 2^s$ , the process of generating  $z$  until  $z < w$  produces a random variable  $z^-$  that takes on any value from  $-0, 1, \dots, w - 1$ -with probability  $1/w$ . Since  $w > 2^{s-1}$ , less than 2 iterations are required on average.

Define

$$\Theta_1 = 0, \quad \Theta_i = \sum_{j < i} \nu_j, \quad i = 2, 3, \dots, n+1, \quad (\Theta_{n+1} = w).$$

Consider a random variable  $\xi$  over  $-0, 1, \dots, w - 1$ -such that  $\xi = i$  iff  $\Theta_i - z^- < \Theta_{i+1}$ . It can be easily found that

$$\Pr -\xi = i - = \Pr -\Theta_i - z^- < \Theta_{i+1} - = \frac{\Theta_{i+1} - \Theta_i}{w} = \frac{\nu_i}{w}.$$

This is our intended distribution (4).

To implement this procedure notice that finding  $\Theta_i$  and  $\Theta_{i+1}$  given  $z^-$  is the same process as finding an interval and a correspondent symbol in arithmetic decoding (which has been described in Sect. 2). Hence, this procedure may be efficiently performed by using the data structure  $\mathbf{C}^-$  defined in Sect. 2. The only difference is that the values  $\nu_i$  are less by one than counters  $c_i$  involved in the construction of  $\mathbf{C}^-$ . This may be easily checked by subtracting  $2^{k-1}$  from  $C_j^k$

in the process of finding a relevant interval. So, generating  $\xi$  does not increase the order of complexity of the method.

The results obtained in this section are summed up in the following

**Theorem 3** *Let there be given a memoryless source generating letters over the alphabet  $A = -a_1, a_2, \dots, a_n-$ . Let the adaptive scheme based on imaginary sliding window and arithmetic coding be applied for encoding symbols generated by the source. Then the memory size of the encoder (decoder)  $S$  and encoding (decoding) time  $T$  are given by the estimates*

$$S < \text{const} \left( n \log \frac{n}{r} \right) , \quad (13)$$

$$T < \text{const} \left( \log \frac{n}{r} \left( \log n + \log \log \frac{n}{r} \log \log \log \frac{n}{r} \right) \right) . \quad (14)$$

So the time complexity of adaptive coding using imaginary sliding window is essentially the same as in case of real sliding window, while the space complexity is less in order of magnitude because the window is not stored (we eliminate the term  $(n/r) \log n$  in (2)).

Let us give some remarks, without any strict considerations, about the ways of obtaining random bits  $z_1 z_2 \dots z_k \dots$ . To maintain the imaginary sliding window these bits are to be known to both the encoder and decoder. A usual way to solve the problem is to use synchronized generators of pseudorandom numbers. We suggest an additional way, namely, using the bits of the code sequence (maybe, with some simple randomizing permutations). The ground for this proposal is that the compressed data are almost random, more exactly, the code sequence approaches a sequence of uniformly distributed code letters as the coding redundancy decreases. The encoder and decoder may use a part of already encoded message for producing random numbers. It is important that the code sequence is known to both the encoder and decoder and they need to store only a small current part of it. This way of obtaining random numbers may occur to be easier than using separate generators.

## References

1. Jelinek, F.: Probabilistic Information Theory. New York: McGraw-Hill (1968) 476–489
2. Rissanen, J. J.: Generalized Kraft inequality and arithmetic coding. IBM J. Res. Dev. **20** (1976) 198–203
3. Pasco, R.: Source coding algorithm for fast data compression. Ph. D. thesis, Dept. Elect. Eng., Stanford Univ., Stanford, CA (1976)
4. Rubin, F.: Arithmetic stream coding using fixed precision registers. IEEE Trans. Inform. Theory **25**, **6** (1979) 672–675
5. Rissanen, J. J., Langdon, G. G.: Arithmetic coding. IBM J. Res. Dev. **23**, **2** (1979) 149–162
6. Guazzo, M.: A general minimum-redundancy source-coding algorithm. IEEE Trans. Inform. Theory **26**, **1** (1980) 15–25

7. Witten, I. H., Neal, R., Cleary, J. G.: Arithmetic coding for data compression. *Comm. ACM* **30**, **6** (1987) 520–540
8. Ryabko, B. Y., Fionov, A. N.: Homophonic coding with logarithmic memory size. *Algorithms and Computation*. Berlin: Springer (1997) 253–262 (Lecture notes in comput. sci.: Vol. 1350)
9. Rissanen J., Langdon G. G.: Universal modeling and coding. *IEEE Trans. Inform. Theory* **27**, **1** (1981) 12–23
10. Cleary, J. G., Witten, I. H.: Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.* **32**, **4** (1984) 396–402
11. Moffat, A.: A note on the PPM data compression algorithm. Res. Rep. 88/7, Dep. Comput. Sci., Univ. of Melbourne, Australia, 1988.
12. Willems, F. M. J., Shtarkov, Y. M., Tjalkens, T. J.: The context-tree weighting method: Basic properties. *IEEE Trans. Inform. Theory* **41**, **3** (1995) 653–664
13. Ryabko, B. Y.: The imaginary sliding window. *IEEE Int. Symp. on Information Theory*. Ulm (1997) 63
14. Krichevsky, R.: *Universal Compression and Retrieval*. Dordrecht: Kluwer Academic Publishers (1994)
15. Feller, W.: *An Introduction to Probability Theory and Its Applications*. New York: Wiley & Sons (1970)

# Robust Protocol for Generating Shared RSA Parameters

Ari Moesriami Barmawi, Shingo Takada, and Norihisa Doi

Department of Computer Science,  
Graduate School of Science and Technology, Keio University,  
3-14-1, Hiyoshi, Yokohama 223, Japan  
{ari, michigan, doi}@doi.cs.keio.ac.jp

**Abstract.** This paper describes how  $n$  parties can jointly generate the parameters for the RSA encryption system while being robust to prevent attacks from cheaters and malicious parties. The proposed protocol generates a public modulus number, without the parties knowing the factorization of that number. Our proposed protocol is similar to that of Boneh-Franklin's protocol. However, when there are two communicating parties our proposed protocol does not need the help of a third party. By using our proposed protocol, we can detect the presence of malicious parties and cheaters among the authorized user. An analysis shows that our proposed protocol has less computational complexity than the protocol of Frankel-MacKenzie-Yung.

## 1 Introduction

There are several cryptographic protocols that require an RSA modulus number for which none of the communicating parties know the factorization (such as [6],[7],[8],[9]). Therefore, it becomes necessary for the parties to jointly generate the RSA parameters (i.e. modulus number, public key and secret key). Boneh and Franklin have proposed a protocol for generating shared RSA parameters [1]. However, their protocol is weak against **malicious parties** (i.e., attackers who can view the servers' memory at any moment, hear the messages which are broadcast and inject his own messages [2]) and **cheaters** (i.e., authorized parties who cheat during the protocol, e.g., cheating which cause a non-RSA modulus to be incorrectly accepted and resulting in the factor of the modulus number being found [4].)

To overcome this problem, many robust protocols for generating RSA parameters have been proposed such as those of Frankel-MacKenzie-Yung [2] and Malkin-Wu-Boneh [4]. These protocols have a quite high computational and communication complexity.

We propose a protocol for generating shared RSA parameters among parties. Our protocol is similar to that of Boneh-Franklin's but when there are two communicating parties, it does not require any help of a third party. Our protocol also takes into account robustness against malicious parties and cheaters, with computational complexity lower than Frankel-MacKenzie-Yung's.

Section 2 describes, our proposed protocol. Section 3 then gives an analysis in terms of security. Section 4 compares our approach with previous ones and section 5 makes concluding remarks.

## 2 The Proposed Protocol

This section gives an overview of our protocol and the details of each procedure, i.e., the generation of a modulus number, the generation of the shared keys and the recovery of the encrypted message.

### 2.1 Overview

For simplicity, we will call party  $P_i$  as  $P_i$ . Suppose that all parties  $P_i$  (for  $i = 1, \dots, n$ ) wish to generate shared RSA parameters. After the execution of our proposed protocol, the RSA modulus number  $N$  and the encryption key  $e$  are publicly known, while the decryption key  $d$  will be shared among the parties in a way which enables threshold decryption. All parties should be convinced that  $N$  is indeed a product of two prime numbers, but neither party knows the factorization of  $N$ .

Our protocol is based on the procedure proposed by Boneh and Franklin. Besides, it is also similar to Cocks' [5], but

we have extended Cocks' algorithm to improve computational efficiency and generalized it to handle more than two parties without weakening the protocol's security.

Our proposed protocol consists of the following three procedures:

1. The generation of RSA modulus number  $N$  (where  $N$  is a product of two prime numbers  $p$  and  $q$ ). No party knows the factorization of  $N$ .
2. The generation of shared decryption key  $d$  for a given encryption key  $e$ .
3. The recovery of an encrypted message.

### 2.2 The Generation of Modulus Number $N$

Our proposed protocol determines  $N$  based on a set of secret numbers that all parties have. Each party  $P_i$  has two secret numbers  $p_i$  and  $q_i$  and the two prime numbers are defined as  $p = (p_1 + p_2 + \dots + p_n)$  and  $q = (q_1 + q_2 + \dots + q_n)$ .

The procedure for generating the modulus number consists of two sub-procedures:

1. Generation of  $p$ .
2. Primality testing of  $p$ .

The Generation of  $N$  is performed based on Protocol IA (Figure 1) as follows:

1. All parties have to agree on the length of modulus number  $N$  in advance.
2. Each party  $P_i$  chooses three large prime numbers  $p_i$ ,  $q_i$  and  $r_i$  (where the size of  $p_i$  is a few digits greater than the size of  $q_i$  and the size of  $q_i$  is greater than a half of the size of  $p_i$ ), a number  $a_i$  which is greater than 2, an odd number  $b_i$  where  $\gcd(a_i, b_i) = 1$ , and  $P_i$ 's secret numbers  $p_i$  and  $q_i$ . We assume that  $p_i$  and  $q_i$  must be congruent to 3 mod 4. Furthermore, each party  $P_i$ :

- First, calculates  $(1) = ( ) ( )^{1-i} ( )^{-i} \bmod$  and then  $(1) - ( ) (1) \bmod$
  - Calculates  $(2) = ( ) ( )^{1-i} ( )^{-i} \bmod$  and then  $(2) - ( ) (2) \bmod$
  - Calculates  $(3) = ( ) ( )^{-i} \bmod$  and then  $(3) - ( ) (3) \bmod$
- Then, calculates  $( (1) )$ ,  $( (2) )$  and  $( (3) )$ . If those values have at least two large prime factors then broadcasts  $(1)$   $(2)$  and  $(3)$  along with . Otherwise has to choose another and repeat step 2.
3. Each party  $+1$  sends
 
$$(1) - ( (1) +1 + (2) +1 + (3) ( +1 +1 )) \bmod$$
  4. Each party broadcasts  $(2) = (1) + (2) \bmod$  for  $= 1$   $2$  and  $= -1$  (e.g. suppose  $= 5$ ,  $= 5$ , then  $_5$  has to broadcast  $_1 5(2)$ ,  $_2 5(2)$ ,  $_3 5(2)$ ).
  5. Each party sends  $(3)$  where

$$(3) = (3) [ ( +1 + +1 + +1 +1 ) + \sum_{\substack{k=1; \\ k \neq i, i+1, \dots, j, j+1}} ( + ) ] \bmod$$

for  $= 1$   $2$   $= +1$  (e.g. if  $= 2$ ,  $= 5$ , then  $= 1$   $4$   $5$ . This means that  $_2$  has to send  $_1 2(3)$  to  $_1$ ,  $_4 2(3)$  to  $_4$ ,  $_5 2(3)$  to  $_5$ ).

6. Each party can calculate the RSA modulus number using the following equation:

$$- ( )^{-1} ( )^i \bmod [ [ (1) + \sum_{\substack{j=1; \\ j \neq i, i+1}} (2) + \sum_{\substack{j=1; \\ j \neq i-1, i}} (3) ] ] \bmod ( ( ) ) + \bmod \quad (1)$$

The example for calculating will be described in Appendix A.

7. Finally each party calculates  $( )$  (where is any hash function that has been agreed upon by all parties in advance) and then broadcasts it.
8. Each party then compares the  $( )$  sent by other parties and the  $( )$  that he just calculated. If all values are equivalent, then all parties will agree on as their RSA modulus number. Otherwise, they will repeat the procedure.

After generating the modulus number , all parties have to test whether is a product of two primes and and to prove that these primes are those used in the primality test to detect the presence of a cheater and/or a malicious party. We use the primality test proposed by Boneh and Franklin [1] with additional steps for detecting cheaters and malicious parties. The procedure is as follows:

1. All parties agree on a number , where the Jacobi symbol of over or must be 1, i.e.,  $( ) = 1$ .

Message 1.  $U_i \rightarrow * : X_i Y_i \delta_i, F_{i(1)}, F_{i(2)}, F_{i(3)}$

Message 2.  $U_{i+1} \rightarrow U_i : Z_{i(1)}$

Message 3.  $U_j \rightarrow * : Z_{i,j(2)}$

Message 4.  $U_j \rightarrow U_i : Z_{i,j(3)}$

Message 5.  $U_i \rightarrow * : H(N)$

**Fig. 1.** Protocol IA

2. Each party calculates

$$- (1) = (x_i - y_i - 1) \bmod N,$$

$$- (2) = (x_i + y_i) \bmod N,$$

and exchanges these results.

Each party can verify whether a party is malicious using the following procedure:

– Each party chooses any integer and then calculates

$$= (x_{j(1)} + x_{j(2)}) \bmod N \quad (2)$$

where the value of (1) and (2) is shown in step 2 of Protocol IA.

Then, he broadcasts for  $i = 1, 2, \dots$ ;  $=$ .

– Each user calculates

$$= (x_j)^{-1} (x_j)^{n_j} \bmod N \quad (3)$$

and

$$= (x_{(2)})^4 \bmod N \quad (4)$$

and broadcasts for  $i = 1, 2, \dots$ ;  $=$ , along with.

– Finally, each user can verify whether is malicious or not using the following expression:

$$= (x_j)^i \bmod N \quad (5)$$

If this expression is NOT TRUE then is a malicious party or a cheater.

If there is a malicious party or a cheater among the parties, then the protocol for modulus number generation should be repeated without including the malicious party or the cheater.

3. Furthermore, they verify that is a product of two primes using the following equation:

$$(1) = -[\prod_{\substack{j=1; \\ j \neq i}}^{} (2)] \bmod (3) \quad (6)$$

Each party may execute equation (6) for  $i = 1, 2, \dots, n$ .

4. If we consider  $i = 1$  where  $(1) = (1)^{-1}$ ,  $(2) = (2)^{-2}$  and  $(3) = -1 \bmod (3)^{(1-1)}$ , then the above two steps will pass *incorrectly*. We thus have to check whether  $(1) + (2) + (3) = 1$ .

For calculating  $(1) + (2) + (3)$ , we will use protocol IB<sup>1</sup> (see figure 2). Each party  $i$  first chooses his own number  $U_i$ . Then they execute Protocol IB.

Message 1.  $U_i \rightarrow * : G_{i(1)}, G_{i(2)}, G_{i(3)}$

Message 2.  $U_{i+1} \rightarrow U_i : V_{i(1)}$

Message 3.  $U_j \rightarrow * : V_{i,j(2)}$

Message 4.  $U_j \rightarrow U_i : V_{i,j(3)}$

Message 5.  $U_i \rightarrow * : H(w)$

**Fig. 2.** Protocol IB

The definition of all variables used in Protocol IB is given below:

- $(1) = (1) + (2) + (3)$  (except for  $i = 1$ ,  $(1) = (1) + (2) - 1$ )
- $(1) = (1) \cdot (2) \cdot (3)^{1-i} \cdot (4)^{-i} \bmod (5)$
- $(2) = (1) \cdot (2) \cdot (3)^{1-i} \cdot (4)^{-i} \bmod (5)$
- $(3) = (1) \cdot (2) \cdot (3)^{-i} \bmod (5)$
- $(1) = (1) \cdot (1) + 1 + (2) \cdot (1) + 1 + (3) \cdot (1) + 1 \bmod (5)$
- $(2) = (1) \cdot (1) + 1 + (2) \cdot (1) + 1 \bmod (5)$
- $(3) = (3) \cdot [(1) \cdot (1) + 1 + (2) \cdot (1) + 1 + (3) \cdot (1) + 1] + \sum_{\substack{k=1; \\ k \neq i, i+1, \dots, j+1}}^{n-1} (1) \cdot (2) \cdot (3) \bmod (5)$
- for  $i, j, k$ , and  $l$  all parties can use the values used in Protocol IA with  $H(\cdot)$  is the hash of  $\cdot$ .

The execution of Protocol IB results in all parties jointly calculating

$$(1) = (1)^{-1} \cdot (2)^{-i} \cdot (3)^{-i} \bmod (5) \quad \left[ [(1) + \sum_{\substack{j=1; \\ j \neq i, i+1}}^{} (2) + \sum_{\substack{j=1; \\ j \neq i-1, i}}^{} (3)] \right] \bmod (5) + \dots \bmod (5)$$

<sup>1</sup> Details of Protocol IB is similar to Protocol IA, and will be omitted due to space.

Then calculating  $x_i = m_i \bmod n$ . According to Boneh-Franklin [1],  $(x_i) = (m_i)$ . Thus, if all parties can verify that  $(x_i) = 1$ , they will reject this number.

Unfortunately, this test will also eliminate a few valid numbers i.e. moduli  $n = p \cdot q$ , for  $p = 1 \bmod n$ . We do not describe the protocol for testing whether  $n$  is a valid number, but it is similar with the test for two parties which was described in [12].

### 2.3 The Generation of Shared Public/Secret Keys

We now describe the procedure for generating public/secret keys. Suppose all parties have successfully calculated  $n$ . Then, they will jointly generate shared decryption key  $d$ , where  $d = \sum_{i=1}^n e_i$  and  $d = e_i^{-1} \bmod (n)$  for some agreed upon value of  $e_i$ . Each party  $P_i$  will have its own decryption exponent  $e_i$ . For generating the keys, we will use the method proposed by Boneh and Franklin [1] but without the help of a third party.

The procedure for generating shared keys are as follows:

- Each party  $P_i$  broadcasts  $(e_i + 1) \bmod n$ .
- Each party calculates  $d = (n) \bmod$  which is congruent to  $[(\sum_{i=1}^n e_i) + (\sum_{i=1}^n e_i) - n] \bmod n$ .
- Since  $(n)$  and  $d$  are relatively prime, then all parties can calculate  $\gamma = -(n)^{-1} \bmod n$ .
- Thus,  $d = 1 + (n)(\gamma)$  and

$$d = \frac{1 + (n + 1 - (\sum_{i=1}^n (e_i + 1))) (\gamma)}{(7)}$$

Each party's decryption key  $d$ , is calculated according to the following procedure:

- Let  $\gamma(n + 1) = \gamma(n + 1) \bmod n = d \bmod n$ . Each party  $P_i$  (except for  $i = 1$ ) broadcasts  $e_i$ .  $P_1$  broadcasts  $e_1 = \gamma(n + 1) - \gamma(n + 1) \bmod n$ .
- Each party  $P_i$  calculates

$$d = e_1 + \sum_{i=1}^n (\gamma(n + 1) - \gamma(n + 1)) \bmod n = \sum_{i=1}^n e_i \bmod n$$

- If  $n = 0$  then execute the following procedure 1:
  1.  $P_1$  calculates  $e_1 = \frac{-(n + 1)\gamma - \gamma(n + 1)}{n}$ .
  2. For  $i = 2$  to  $n$ ,  $P_i$  calculates  $e_i = \frac{(n + 1)\gamma - \gamma(n + i)}{n}$ .
  3. For  $i = n$  to  $2$ ,  $P_i$  calculates  $e_i = \frac{(n + 1)\gamma - \gamma(n + i)}{n}$ .
- If  $n \neq 0$  then
  1.  $P_1$  executes step 1 of procedure 1.
  2. For  $i = 2$  to  $n - 2$ , execute step 2 of procedure 1.
  3. For  $i = n - 2$  to  $n$ , execute step 3 of procedure 1.

For verifying that the key generation is done successfully, all parties have to agree on a message  $m$  in advance, then each party has to sign this message using his own decryption key and broadcast the signed message. Furthermore, each party multiplies all signed messages and decrypts it with the public key. th

### 3 Security Analysis

In this section we will analyze the ability to calculate the factors of modulus number  $N$ , and also summarize the security requirements.

#### 3.1 Ability for Finding Factors of $N$

Since the strength of our proposed protocol is on breaking the modulus number  $N$ , we will analyze how far the messages which a party obtained will leak information for obtaining other party's secret number  $x$  and  $y$ .

First, we will analyze each message of the protocol for generating  $z$  (Protocol IA). Assume that each party only saves  $z$ , while  $x$ ,  $y$ ,  $r$ , and  $s$  have to be destroyed or changed after a round of executing Protocol IA. Message 1 contains numbers which are functions of  $x$ ,  $y$ ,  $r$ ,  $s$ ,  $N$ , and  $z$ . There are three ways to obtain the values of  $x$  and  $y$  by other parties:

- Other parties can obtain  $x$  and  $y$  iff they can find the inverse of the function  $f_{(3)}$  since the product of  $((f_{(3)})^{-1} \bmod N)$  and  $(f_{(1)} \bmod N)$  is  $(x \bmod N)$  and the product of  $((f_{(3)})^{-1} \bmod N)$  and  $(f_{(2)} \bmod N)$  is  $(y \bmod N)$ . There is no possibility of calculating the multiplicative inverse of  $f_{(3)}$  since  $x$  and  $y$  are not relatively prime.
- Other parties can obtain  $x$  and  $y$  by finding the factors of  $N$  because if they knew  $N$ , then they can find  $(f_{(1)} \bmod N)$ ,  $(f_{(2)} \bmod N)$  and  $(f_{(3)} \bmod N)$ . This means that they can find the multiplicative inverse of  $(f_{(3)} \bmod N)$ . Thus, they can obtain  $x$  and  $y$  by multiplying  $(f_{(1)} \bmod N)$  and  $(f_{(2)} \bmod N)$  with the multiplicative inverse of  $(f_{(3)} \bmod N)$ . As described above, only  $z$  is saved by  $P_1$ . Thus, the possibility of obtaining  $x$  and  $y$  is equal to the possibility of factoring  $N$ .
- $x$  and  $y$  can also be obtained by calculating  $(f_{(1)} \bmod N)$ ,  $(f_{(2)} \bmod N)$  and  $(f_{(3)} \bmod N)$ . From these processes they can obtain  $x$  and furthermore obtain  $y$ . But, since  $f_{(1)}$ ,  $f_{(2)}$  and  $f_{(3)}$  have at least two large prime factors, it is still hard to obtain  $x$  and  $y$ .

Since messages 2, 3 and 4 contain functions which have more than three unknown variables, then there are no possibility of obtaining  $x$  and  $y$  from these messages.

Even if a malicious party or a cheater who can corrupt  $z$  is still difficult for them to pass our proposed protocol correctly, since to pass the protocol they have to find the factors of  $N$  which is not saved in the memory of  $P_1$ .

#### 3.2 Security Requirement

Based on the above discussion, there are a few conditions that need to be satisfied to keep the factors of modulus number  $N$  secret. These conditions can be summarized as follows:

- All parties have to determine the length of modulus number  $n$  in advance.
- Each party  $P_i$  has to choose the length of  $p_i$  to be a few digits longer than the length of  $n$ .
- Each party  $P_i$  has to choose the length of  $p_i$  and  $q_i$  such that the length of  $n$  is about the length of  $p_i$ .

## 4 Comparison with Other Protocol

Protocols have been proposed for jointly generating RSA parameters, such as Boneh-Franklin and Frankel-MacKenzie-Yung for  $n$  parties and Cocks and Poupard-Stern [3] for two parties.

Since our proposed protocol is for  $n$  parties, then we will compare it with the protocol of Boneh-Franklin [1] and Frankel-MacKenzie-Yung [2].

The benefit of our proposed protocol compared with Boneh-Franklin is that when there are two communicating parties, our protocol does not need any third party for calculating the modulus number  $n$  as well as the keys.

The probability of generating an RSA modulus number from two random primes of  $k$  bits each is about  $(2^{-k})^2$ , which means that we will have about  $2^{2k}$  rounds. Each round will have computational complexity about  $12k$  modular exponentiations and communication complexity is about  $(10k - 3)$ . Thus, the computational complexity is about less than a third compared with the protocol of Frankel-MacKenzie-Yung (which is  $24k + 1$ ). Another benefit of our proposed protocol is that the communication complexity does not depend on the size of  $k$ .

## 5 Conclusion

We have proposed a protocol for jointly generating parameters in RSA encryption. When there are two communicating parties, our protocol does not need the help of a third party, and it has less computational complexity compared with previous protocols. The advantage of our proposed protocol is that the communication complexity does not depend on the size of  $k$ .

## References

1. Boneh, D. and Franklin, M.: Efficient Generation of Shared RSA Keys. *Advances in Cryptology-Crypto '97. Lecture Notes in Computer Science*. Springer Verlag (1997), 423-439.
2. Frankel, Y., MacKenzie, P. D., Yung, M.: Robust Efficient Distributed RSA-Key Generation. *Proceedings STOC 98. ACM* (1998).
3. Poupard, G. and Stern, J.: Generation of Shared RSA Keys by Two Parties. *Proceedings of ASIACRYPT 98. Lecture Notes in Computer Science*. Springer Verlag (1998), 11-24.
4. Malkin, M., Wu, T. and Boneh, D.: Experimenting with Shared Generation of RSA Keys. *Proceedings of Internet Society's 1999 Symposium on Network and Distributed Network Security*, (1999).

5. Cocks, C.: Split Knowledge Generation of RSA Parameters. Proceedings of 6th International Conference of IMA on Cryptography and Coding. Lecture Notes in Computer Science. Springer Verlag (1997), 89-95.
6. Feige, U., Fiat, A. and Shamir, A.: Zero-knowledge Proofs of Identity. Journal of Cryptology, 1, (1988), 77-94.
7. Fiat, A. and Shamir, A.: How to Prove Yourself: Practical Solution to Identification Problems. Crypto '86. Lecture Notes in Computer Science (1986), 186-194.
8. Ohta, K. and Okamoto, T.: A modification of Fiat-Shamir scheme. Crypto '88. Lecture Notes in Computer Science. Springer Verlag (1988), 232-243.
9. Ong, H. and Schnorr, C.: Fast Signature Generation with a Fiat-Shamir-like Scheme. Eurocrypt '90. Lecture Notes of Computer Science. Springer Verlag (1990), 432-440.
10. Rivest, R. L., Shamir, A. and Adleman, L.: Method for Obtaining Signatures and Public-Key Cryptosystems. Communication of the ACM (1978).
11. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley and Sons (1994).
12. Barmawi, A. M., Takada, S., Doi, N.: A Proposal for Generating Shared RSA Parameters. Proceeding of IWSEC 99, September, 1999.

## A Calculating $N$ by Each User

Here we will simulate how a party can calculate the modulus number using all information that it has obtained. Suppose  $a = 1$ ,  $b = 5$ , then  $U_1$  can calculate using equation (1) as follows:

$$\begin{aligned}
 & - (a^{-1})^{-1} (a^{-1})^{-1} a^{-1} \bmod a^{-1} [ [ a^{-1}(1) + \sum_{\substack{j=1; \\ j \neq 1,2}}^5 a^{-1}(2) + \sum_{\substack{j=1; \\ j \neq 1,5}}^5 a^{-1}(3) ] a^{-1} \\
 & \bmod ((a^{-1} a^{-1}) a^{-1}) + a^{-1} a^{-1} \bmod a^{-1} a^{-1} \quad (8)
 \end{aligned}$$

Let  $U_1 = (a^{-1})^{-1} (a^{-1})^{-1} a^{-1} a^{-1}(1)$ ,  $U_2 = (a^{-1})^{-1} (a^{-1})^{-1} a^{-1} (\sum_{i=1+2}^5 a^{-1}(2))$ , and  $U_3 = (a^{-1})^{-1} (a^{-1})^{-1} a^{-1} (\sum_{i=2}^4 a^{-1}(3))$ .

Then, equation 8 can be written as:

$$= U_1 + U_2 + U_3 \bmod a^{-1} a^{-1} \quad (9)$$

Next, we will calculate  $U_1$ ,  $U_2$  and  $U_3$  in detail.

$$\begin{aligned}
 U_1 &= (a^{-1})^{-1} (a^{-1})^{-1} (a^{-1})^{-1} a^{-1}(1) \\
 &= (a^{-1})^{-1} (a^{-1})^{-1} (a^{-1})^{-1} [ a^{-1}(1)^{1-1} (a^{-1})^{-1} a^{-1} 2 + a^{-1}(a^{-1})^{-1} (a^{-1})^{1-1} a^{-1} 2 + \\
 &\quad a^{-1}(a^{-1})^{-1} (a^{-1})^{-1} a^{-1} 2 ] \\
 &= a^{-1} 2 + a^{-1} 2 + a^{-1} 2 \quad (10)
 \end{aligned}$$

$$\begin{aligned}
_2 &= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} \left[ \sum_{\substack{j=1; \\ j \neq 1,2}}^{=5} {}_1({}_2) \right] \\
&= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} \left[ \sum_{=3}^{=5} {}_1({}_2) \right] \\
&= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} [{}_1{}_3(2) + {}_1{}_4(2) + {}_1{}_5(2)] \\
&= {}_1{}_3 + {}_3{}_1 + {}_1{}_4 + {}_1{}_4 + {}_1{}_5 + {}_5{}_1 \quad (11)
\end{aligned}$$

$$\begin{aligned}
_3 &= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} \left[ \sum_{\substack{j=1; \\ j \neq 1,5}}^{=5} {}_1({}_3) \right] \\
&= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} \left[ \sum_{=2}^{=4} {}_1({}_3) \right] \\
&= ({}_1)^{-1}({}_1)^{-1}({}_1)^{-1} [{}_1{}_1^{-1}{}_1^{-1}{}_1^{-1}] [-{}_2{}_3 + {}_3{}_2 + {}_3{}_3 + \sum_{\substack{k=1; \\ k \neq 1,2,3}}^{=5} {}_2 + {}_2 \\
&\quad + -{}_3{}_4 + {}_4{}_3 + {}_4{}_4 + \sum_{\substack{k=1; \\ k \neq 1,2,3,4}}^{=5} {}_3 + {}_3- \\
&\quad + -{}_4{}_5 + {}_5{}_4 + {}_5{}_5 + \sum_{\substack{k=1; \\ k \neq 1,2,3,4,5}}^{=5} {}_4 + {}_4- \\
&= {}_2{}_3 + {}_3{}_2 + {}_3{}_3 + {}_2{}_4 + {}_4{}_2 + {}_2{}_5 + {}_5{}_2 + {}_3{}_4 + {}_4{}_3 + {}_4{}_4 \\
&\quad + {}_3{}_5 + {}_5{}_3 + {}_4{}_5 + {}_5{}_4 + {}_5{}_5 \quad (12)
\end{aligned}$$

Thus, by using equations (10), (11), and (12), we can obtain:

$$\begin{aligned}
&= {}_1 + {}_2 + {}_3 \bmod {}_1{}_1 \\
&= [{}_1{}_2 + {}_2{}_1 + {}_2{}_2 + {}_1{}_3 + {}_3{}_1 + {}_1{}_4 + {}_1{}_4 + {}_1{}_5 + {}_5{}_1 + {}_2{}_3 \\
&\quad + {}_3{}_2 + {}_3{}_3 + {}_2{}_4 + {}_4{}_2 + {}_2{}_5 + {}_5{}_2 + {}_3{}_4 + {}_4{}_3 + {}_4{}_4 + {}_3{}_5 \\
&\quad + {}_5{}_3 + {}_4{}_5 + {}_5{}_4 + {}_5{}_5 + {}_1{}_1] \bmod {}_1{}_1 \\
&= ({}_1 + {}_2 + {}_3 + {}_4 + {}_5)({}_1 + {}_2 + {}_3 + {}_4 + {}_5) \bmod {}_1{}_1 \quad (13)
\end{aligned}$$

# Some Soft-Decision Decoding Algorithms for Reed-Solomon Codes

Stephan Wesemeyer<sup>\*</sup>, Peter Sweeney, and David R.B. Burgess

Centre for Comm. Systems Research, University of Surrey, Guildford GU2 5XH, U.K.

**Abstract.** In this paper we introduce three soft-decision decoding algorithms for Reed-Solomon (RS) codes. We compare them in terms of performance over both the AWGN and Rayleigh Fading Channels and in terms of complexity with a special emphasis on RS codes over  $\mathbb{F}_{16}$ . The algorithms discussed are variants of well known algorithms for binary codes adapted to the multilevel nature of RS codes. All involve a re-ordering of the received symbols according to some reliability measure. The choice of reliability measure for our simulations is based on a comparison of three in terms of how they affect the codes' performances.

## 1 Introduction

It is well known that one way of facilitating soft-decision decoding for linear block codes is to represent them by a trellis and apply the Viterbi algorithm (VA) to decode them. However, the complexity of the VA makes its use infeasible for all but a small number of linear codes. Because of the widespread use of RS codes, it would be highly desirable to find efficient soft-decision algorithms for them. Various approaches have been proposed (see [1] for a recent example). This paper introduces a further three. Our simulations were based around an AWGN and a Rayleigh fading channel with BPSK (binary-phase-shift-keyed) modulation and 8-level uniform quantisation. Except in a very few cases with extremely long simulation runs, we based the results on 100 error events (word errors, not bit errors). Throughout the paper we denote by  $\mathbb{F}_q$ , a finite field of  $q = 2^l$  elements and assume an  $[n, k]$  linear code over  $\mathbb{F}_q$  which can correct  $t$  errors.

## 2 The Algorithms

The Dorsch algorithm was proposed in [2] for binary codes and has more recently been applied by Fossorier and Lin [3]. Given a code of length  $n$  and dimension  $k$  the idea is to find  $k$  most reliable symbols whose positions are such that they can be used as an information set of the code. Various error patterns are added to this information set and each result is re-encoded. In each case, the distance of the obtained codeword from the received word is computed. Decoding stops as soon as we have a maximum-likelihood solution or the number of permitted decoding

---

<sup>\*</sup> The research was supported by an EPSRC grant.

tries has been exhausted (in which case the best solution up to that point is output). Our first two algorithms (A1 and A2) are based on this technique.

The codeword closest to the received word in terms of the following metric is the maximum-likelihood solution we want to find.

**Definition 1.** Let  $s_i = (s_{i1}, \dots, s_{il}) - \mathbb{F}_q$  be the symbol obtained by using hard decision ( $s_{ij} = 0, 7$ ) on  $r_i = (r_{i1}, \dots, r_{il})$ , the  $i$ th received symbol after quantisation. The distance between a received word  $r = (r_1, r_2, \dots, r_n)$  and a word  $c = (c_1, \dots, c_n) - \mathbb{F}_q^n$ , with  $c_i = (c_{i1}, \dots, c_{il}) - \mathbb{F}_q$ , is defined as

$$\text{dist}(r, c) = \sum_{i=1}^n (\text{dist}_{\text{sym}}(r_i, c_i)) \text{ where } \text{dist}_{\text{sym}}(r_i, c_i) = \sum_{j=1}^l \varphi_{ij} - c_{ij} - \sum_{j=1}^l \varphi_{ij} - s_{ij} -$$

Furthermore, all algorithms produce a continuous stream of possible solutions which are subjected to a stopping criterion that, if satisfied, is sufficient (though not necessary) to guarantee a maximum-likelihood solution [4], in which case the decoding stops. Since we are concerned here with RS codes, any  $k$  symbols may be used as an information set. Hence we simply sort the symbols according to reliability (see Chapter 3) and, in algorithm A1, we use the  $k$  most reliable as the information set. A2 repeats A1 using the  $k$  least reliable symbols unless a maximum likelihood solution has already been found by A1.

Fossorier and Lin's implementation of the Dorsch algorithm checks error patterns corresponding to  $i$  errors in the information set. This has been termed order- $i$  reprocessing [3]. In our version, we take a slightly different approach which is closer to the original Dorsch algorithm. Our order for testing the error patterns to be added to the chosen information set is the proximity of the resulting sequence to the corresponding part of the received word. The index used is the generalisation of 'dist' to different length sequences which takes the sum of all the 'dist<sub>sym</sub>'s over the symbols of the sequence. This is achieved by using a stack-type algorithm, whereby stacks of sequences of different lengths are kept in storage, ordered according to the index. A sequence from the stack of lowest index is extended in  $q$  different ways by appending a symbol, the indices of the resulting sequences are calculated and they are each put in the appropriate stack. The memory requirement of this implementation is determined by the maximum number of decoding tries. Let  $MDT$  be this maximum and  $DT$  be the number of decoding tries so far. Then we only need to keep  $MDT - DT$  information sets of smallest index in our array as none of the others will be used.

Our third algorithm (A3) simply applies A1 and, if that algorithm does not produce a maximum-likelihood solution, then a Chase-style algorithm is applied to the sorted word, i.e. we apply a fixed number of error patterns of least distance to the least reliable symbols and then use an algebraic decoder to decode. This approach has already been applied successfully to binary codes by Fossorier and Lin [5].

### 3 Sorting

All the algorithms depend on sorting the received symbols according to some reliability measure. In the case of binary codes, Fosserier and Lin showed that on an AWGN and on a Rayleigh fading channel with BPSK modulation, the absolute value of the received symbol is the most appropriate choice [3]. The higher that value is, the more reliable hard decision on the received symbol will be. With RS or indeed any code whose symbols come from a non-binary finite field we need to find a slightly different approach. In such a case, each 'received symbol' will, in fact, be a string of symbols which, between them, indicate the binary representation of the 'received symbol'.

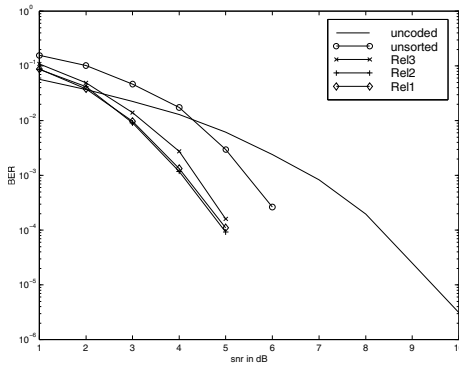
**Definition 2.** Let  $r = (r_1, \dots, r_l)$  with  $0 \leq r_i \leq 7$  be a received symbol after quantisation and define

$$\text{Rel}_1(r) = \sum_{i=1}^l 3.5 - r_i, \quad \text{Rel}_3(r) = \min_{i=1, \dots, l} 3.5 - r_i$$

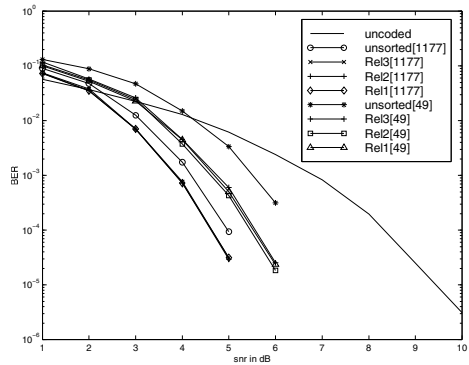
$$\text{and } \text{Rel}_2(r) = \prod_{i=1}^l P(hd(r_i) \neq i), \quad \text{where } hd(j) = \begin{cases} 0 & : 0 \leq j \leq 3 \\ 1 & : 4 \leq j \leq 7 \end{cases}$$

The natural generalisation of the reliability measure of the binary case is to add the absolute values of the symbols in the string, thus obtaining an overall reliability of the 'received symbol'. As we use 8-level uniform quantisation this translates into  $\text{Rel}_1$  above. Another approach is to use Bayes' rule. One can easily determine the probability  $P(j|0)$  (resp.  $P(j|1)$ ) of a received bit being quantised to level  $j$  given that a 0 (resp. a 1) was transmitted. From that we can work out the probability  $P(0|j)$  (resp.  $P(1|j)$ ) that a 0 (resp. 1) was transmitted given that we are in level  $j$ . Hence we arrive at  $\text{Rel}_2$ . Lastly, the most basic approach simply takes the least reliable bit in a symbol and uses its value as the overall reliability of the symbol ( $\text{Rel}_3$ ). These three reliability measures were felt to be the most natural ones. It can easily be seen that the higher the computed reliability of a symbol is, the more likely it is to be correct.

Figure 1 (respectively Figure 2) contains the results for a  $[16, 8, 9]$  ( $[16, 12, 5]$ ) extended RS code over the AWGN channel (see Section 6.2 for the Rayleigh channel results), decoded using algorithm A1 with a maximum number of decoding tries corresponding to the number of order-2 (order-2 and order-1) reprocessing attempts with and without sorting. Note that, to compute the probabilities accurately for  $\text{Rel}_2$ , we need to know at which signal-to-noise ratio (SNR) the bits were transmitted. As this information is not always available in practice, we computed the probabilities for a SNR of 1dB adjusted by the code rate,  $R$  say, i.e.  $\text{SNR} = R - 10^{0.1}$  and used these values throughout. (The simulations showed that - if anything - this approach proved slightly better than using the exact values for the different SNRs.)



**Fig. 1.** Sorted vs unsorted  $[16, 8, 9]$  extended RS code - 529 decoding tries



**Fig. 2.** Sorted vs unsorted  $[16, 12, 5]$  extended RS code - 1177 and 49 decoding tries

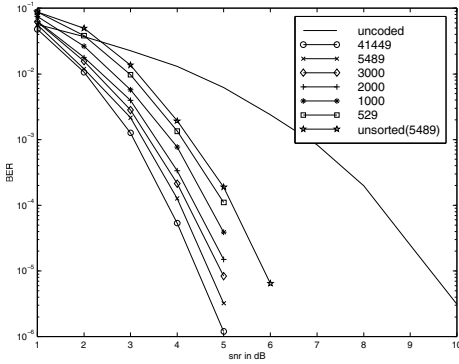
As can be seen all reliability measures result in a marked improvement over the unsorted case. The reason why  $\text{Rel}_3$  is slightly worse than the other two (except in the case of 1177 decoding tries for the higher rate code) can be explained by the observation that the number of different reliability levels attached to each symbol in that method is rather low (4 compared to 26 for  $\text{Rel}_1$  and 35 for  $\text{Rel}_2$ ). At 1177 decoding tries, the algorithm performs close to maximum-likelihood decoding in any case - it is not important whether or not the least distorted symbols are used as an information set.

Because there was no significant difference between sorting the symbols of the received words according to  $\text{Rel}_1$  or  $\text{Rel}_2$  we used sorting by  $\text{Rel}_1$  in all the remaining simulations.

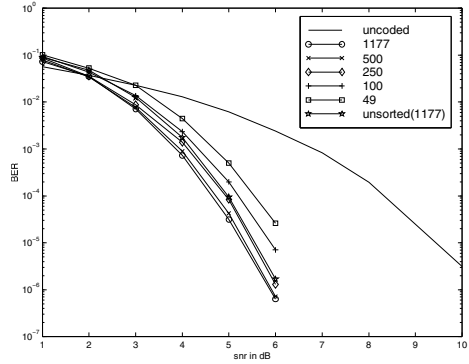
## 4 Number of Decoding Tries

The most crucial feature of the proposed algorithms is the number of decoding tries they entail. The more decoding tries the more likely it is that we find the maximum-likelihood solution. However, as Fossorier and Lin [3] demonstrated the actual gain obtained from further decoding tries has to be measured against the extra computation involved.

Figure 3 is an example of how the maximum number of decoding tries (using algorithm A1) after sorting (with respect to  $\text{Rel}_1$ ) can affect the performance of a code and how this performance compares to the unsorted case. Note that 41449, 5489, and 529 correspond to the number of decoding tries given by order-4, order-3, and order-2 reprocessing respectively. There is a marked improvement of about 1dB going from 529 decoding tries to 5489 but only a slight improvement of roughly 0.25dB when 41449 attempts are used instead of 5489 which does not justify the almost 8-fold increase in number of decoding tries. However, even then the complexity of the proposed algorithm is several orders of magnitude lower



**Fig. 3.** [16, 8, 9] extended RS code - different numbers of permitted decoding tries



**Fig. 4.** [16, 12, 5] extended RS code - different numbers of permitted decoding tries

than that of the Viterbi algorithm which would have to deal with  $16^8 \approx 4.3 \cdot 10^9$  states for this code.

Figure 4 shows the effect of different numbers of permitted decoding tries (using algorithm A1) for a [16, 12, 5] extended RS code. This time we restricted the number of decoding tries to lie in between 49 and 1177 (= number of decoding tries for order-1 and order-2 reprocessing respectively). Note that decoding after sorting with a maximum of 250 decoding tries slightly outperforms unsorted decoding with maximum 1177 decoding tries and there is only a very slight improvement going from 500 to 1177 decoding tries.

## 5 Measures of Complexity

The complexity of each algorithm is expressed in terms of additions, multiplications and comparisons which, for simplicity, are considered equivalent operations. All the estimates we give are based on our implementation; the idea is to give a rough idea of how much computational effort has to be expended on decoding. To enable us to compare the results with other algorithms and to eliminate the code rate as a factor, for each code considered we measure the complexity in operations per information bit. We compare our results throughout with the Viterbi algorithm applied to a convolutional code of rate  $R = 0.5$  and memory  $k = 7$  even though the rate of the RS codes vary. Higher rate convolutional codes are usually obtained by puncturing which does not greatly affect the number of operations which can be estimated at 128 comparisons (= number of states) plus 256 additions (= number of branches).

Our implementation of the A1 and A2 algorithms require, before the re-encoding starts, computing the metric and some values for the stopping criterion ( $n(q - 1)^2$  comparisons and  $nlq$  additions), sorting the symbols according to reliability (approximately  $n \log_2(n)$  comparisons) and reducing a  $(k, n)$  matrix to

reduced echelon form (REF) ( $nk^2$  multiplications and  $nk(k-1)$  additions). This latter is performed twice in  $A2$  (two directions of decoding), so the preliminary operations for algorithm  $Ar$  ( $r=1,2$ ) total:

$$PopAr = n(\log_2(n) + lq + (q-1)^2 + r(k^2 + k(k-1))). \quad (1)$$

For each decoding try (both algorithms), there are the following approximations: re-encoding ( $(n-k)k$  multiplications and  $(n-k)(k-1)$  additions), determining the distance from the received word ( $(n-1)$  additions), determining whether the stopping criterion is satisfied ( $(n+2+n\log_2(n))$  comparisons and  $n-k+1$  additions), determining the best solution (1 comparison per decoding try after the first). Thus altogether the algorithm  $Ar$  ( $r=1,2$ ) requires the following total operations (where  $DT$  is the number of decoding tries).

$$TopAr = PopAr + DT(2(n-k)k + 2n + n\log_2(n) + 2) + (DT-1) \quad (2)$$

The estimates for our implementation of the Chase part of  $A3$  are based on a very general algorithm presented in Stichtenoth [6] and due to A.N.Skorobogatov and S.G.Vladut. The following are the operations per decoding try: Computing the syndrome ( $(n-k)n$  multiplications and  $(n-k)(n-1)$  additions), checking whether the syndrome is 0 ( $n-k$  comparisons), reducing the  $(t, t+1)$  syndrome matrix to REF ( $(t+1)t^2$  multiplications and  $(t+1)t(t-1)$  additions), finding the error locator polynomial ( $t(t+1)/2$  multiplications and the same number of additions), determining the roots of that polynomial (a maximum of  $qt$  multiplications,  $qt$  additions and  $q$  comparisons), finding the error values ( $((t+1)(n-k)^2$  multiplications and  $(t+1)(n-k)(n-k+1)$  additions), obtaining the codeword ( $t$  additions) and computing the distance from the received word and applying the stopping criterion ( $(n-1) + (n-k+1)$  additions and  $(n+2+n\log_2(n))$  comparisons). Thus, denoting by  $DTC$  the number of decoding tries involved in the error-only decoder, the total number of operations required for the  $A3$  algorithm is given by

$$TopA3 = TopA1 + DTC(2(n-k)n + (t+1)(2t^2 + 2q + (n-k)(2(n-k)+1)) + t + 3n - k + 2 + q + n\log_2(n)) \quad (3)$$

As all our algorithms apply a stopping criterion it is easy to see that the higher the SNR, the fewer the decoding attempts needed on average. In our simulation we computed the average number of decoding tries per received word which is then used to compute the total number of operations as given by the above formulae. It is worth noting that the complexity of all three algorithms is dominated by the number of decoding tries. Only for high SNRs, when the average number of decoding tries becomes very small, do the preliminary operations contribute significantly to the average number of operations per information bit.

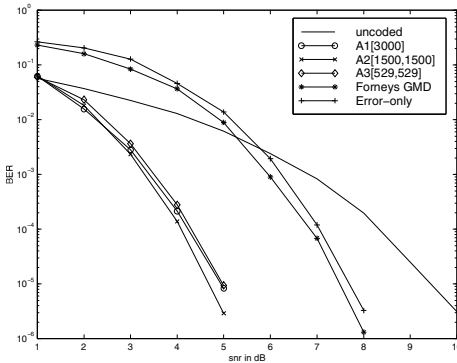
## 6 Comparing the Three Decoding Algorithms in Terms of Performance and Complexity

### 6.1 AWGN Channel

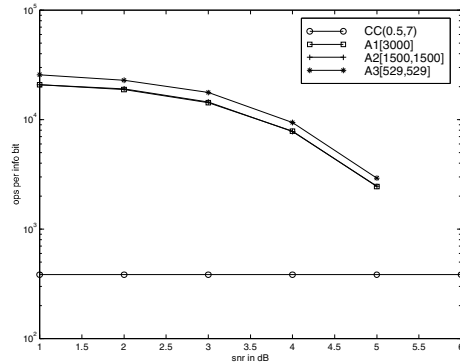
Figures 5, 7 and 9 show the performance of the algorithms when applied to respectively a  $[16, 8, 9]$ , a  $[16, 10, 7]$  and a  $[16, 12, 5]$  extended RS code. The numbers next to the various algorithms indicate the number of permitted decoding tries, e.g. for the  $[16, 8, 9]$  code, the A2 algorithm was run with maximum 1500 decoding tries for each side, and the A3 algorithm was run with 529 (first number) Dorsch-style decoding tries permitted and the same maximum number of Chase-style decoding tries. We have included the performance of Forney's GMD [7] and an error-only decoder to enable the reader to compare the new algorithms with two standard ones. Tables 1, 2 and 3 show how many decoding tries were needed for each algorithm at various SNRs. Figures 6, 8 and 10 show the complexity of the algorithms based on the figures in the tables.

**Table 1.** Ave. num. of decoding tries ( $[16, 8, 9]$  extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB
A1[41449]	40018	36407	27781	14732	4353
A1[3000]	2912	2636	1985	1076	316
A2[1500, 1500]	2902	2666	2006	1063	310
A3[529, 529]	[519, 519]	[462, 462]	[356, 355]	[189, 187]	[57, 55]



**Fig. 5.**  $[16, 8, 9]$  extended RS code decoded using A1, A2, and A3

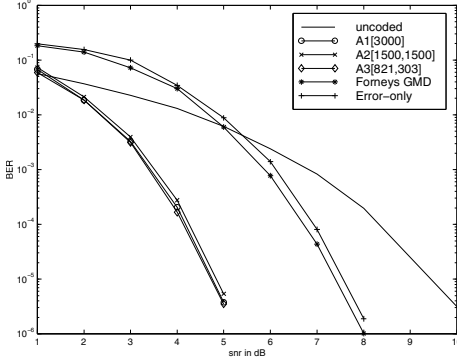


**Fig. 6.** Complexity of the algorithms ( $[16, 8, 9]$  extended RS code)

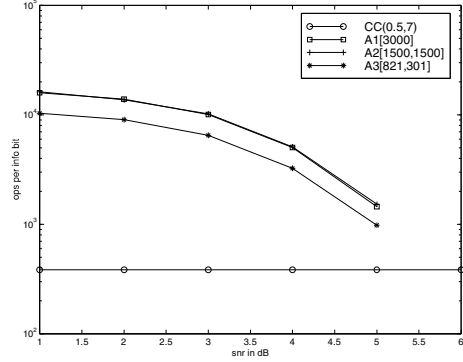
Comparing Figure 5 with Figure 7, in terms of the BER at various SNR there is hardly any difference between the  $[16, 8, 9]$  and the  $[16, 10, 7]$  codes, probably

**Table 2.** Ave. num. of decoding tries ([16, 10, 7] extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB
A1[3000]	2865	2513	1805	882	229
A2[1500, 1500]	2908	2460	1811	883	229
A3[821, 301]	[795, 291]	[693, 254]	[497, 181]	[241, 87]	[64, 22]



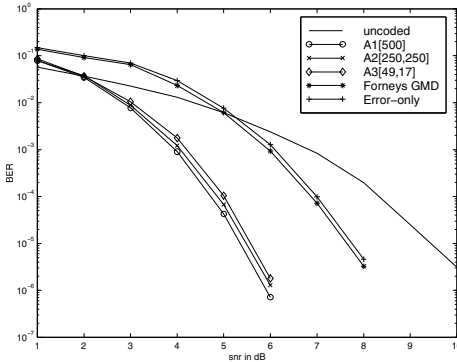
**Fig. 7.** [16, 10, 7] extended RS code de-coded using A1, A2, and A3



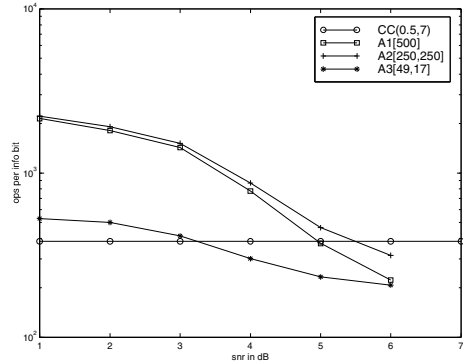
**Fig. 8.** Complexity of the algorithms ([16, 10, 7] extended RS code)

**Table 3.** Ave. num. of decoding tries ([16, 12, 5] extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB	6dB
A1[500]	482	401	305	145	44	7
A2[250, 250]	477	403	304	145	44	7
A3[49, 17]	[46, 15]	[41, 14]	[29, 10]	[16, 5]	[6, 2]	[2, 1]



**Fig. 9.** [16, 12, 5] extended RS code de-coded using A1, A2 and A3



**Fig. 10.** Complexity of the algorithms ([16, 12, 5] extended RS code)

due to the fact that these decoding algorithms are suboptimal and hence do not achieve the full potential of the lower rate code. In addition, in both Figure 6 and Figure 8, the pair of curves “A1[3000]” and “A2[1500, 1500]” overlap. However, whereas the A3 algorithm appears to be the best choice for the [16, 10, 7] code in terms of both performance and complexity, for the rate 1/2 code, A2 slightly outperforms the other two algorithms and (like A3) allows a straightforward parallel implementation, so it is the preferable choice for this code.

In the case of the [16, 12, 5] extended RS code the A1 algorithm performs slightly better than the other two. However, it is worth noting that the A3 algorithm achieves good results with a very low maximum number of decoding tries and that by slightly increasing the number of decoding tries for the A3 algorithm, from [49, 17] to [100, 50], say, one gets a similar performance to the A1 algorithm while still having a lower complexity and the advantage of being able to implement it in parallel. This time even for low SNRs the complexity of A3 is only slightly worse than that of the Viterbi algorithm. At higher SNRs all algorithms achieve good results with few decoding tries resulting in very few operations per information bit.

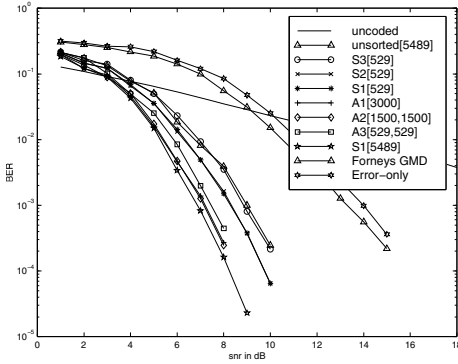
### 6.2 Rayleigh Fading Channel ([16, 8, 9] Extended RS Code Only)

In our simulations we have assumed a perfectly interleaved Rayleigh fading channel, i.e. the fading amplitudes for each bit were completely independent and no channel side information was used in the decoding. We have used the same metric as for the AWGN channel.

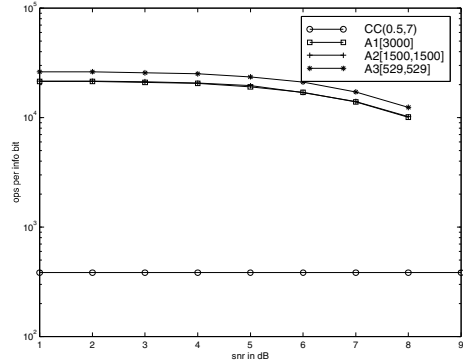
**Table 4.** Ave. num. of decoding tries ([16, 8, 9] extended RS code (Rayleigh channel))

Algorithm	2dB	3dB	4dB	5dB	6dB	7dB	8dB
A1[3000]	3000	2929	2865	2670	2383	1941	1397
A2[1500, 1500]	3000	2964	2879	2726	2359	1944	1400
A3[529, 529]	[529, 529]	[518, 518]	[506, 506]	[475, 475]	[426, 425]	[346, 345]	[250, 248]

Figure 11 shows that the results for the Rayleigh fading channel do not differ very much from the ones we obtained for the AWGN channel. We see, as for AWGN in Section 3, that sorting with respect to the reliability measure  $\text{Rel}_1$  or  $\text{Rel}_2$  (the two curves overlap) is better than  $\text{Rel}_3$ . Furthermore, as in the AWGN channel, sorting with 529 decoding tries yields a better performance than unsorted decoding with 5489 decoding tries. For the Rayleigh fading channel, sorting yields a coding gain of about 2dB when compared to the unsorted case with the same number of decoding tries. This time it seems that the algorithms A1 and A2 perform identically. However, looking at the computed BER values, there is an indication that A2 might outperform A1 slightly for SNRs higher than these. In terms of complexity - see Figure 12 and Table 4 - the only difference



**Fig. 11.**  $[16, 8, 9]$  extended RS code de-coded using A1,A2,A3 (Rayleigh)



**Fig. 12.** Complexity of algorithms  $([16, 8, 9]$  extended RS code (Rayleigh))

from the AWGN channel is that the average number of decoding tries decreases more slowly which is obviously due to the nature of the Rayleigh fading channel. Note that, again, the curves for “A1[3000]” and “A2[1500,1500]” overlap.

## 7 Conclusion

In this paper we have introduced three suboptimal decoding algorithms for RS codes all of which achieve a reduction in complexity of several orders of magnitude over the Viterbi algorithm for these codes whilst keeping the loss in coding gain very small. These algorithms are not restricted to RS codes and could be applied to any linear block code. They achieve their full potential with high rate codes where a small number of decoding tries yields almost maximum-likelihood decoding performance with low decoding complexity.

## References

1. Wesemeyer S. and Sweeney P.: Suboptimal soft-decision decoding for some RS-codes. IEE Electronics Letters **34**(10) (1998) 983–984
2. Dorsch B.G.: A decoding algorithm for binary block codes and J-ary output channels. IEEE Trans. Inform. Theory **IT-20**(3) (1974) 391–394
3. Fossorier M.P.C. and Lin S.: Soft-decision decoding of linear block codes based on ordered statistics. IEEE Trans. Inform. Theory **41**(5) (1995) 1379–1396
4. Taipale D.J. and Pursley M.B.: An improvement to generalized-minimum-distance decoding. IEEE Trans. Inform. Theory **37**(1) (1991) 167–172
5. Fossorier M.P.C. and Lin S.: Complementary reliability-based decodings of binary linear block codes. IEEE Trans. Inform. Theory **43**(5) (1997) 1667–1672
6. Stichtenoth, H.: Algebraic function fields and codes. Springer-Verlag, 1993
7. Forney Jr, G.D.: Generalized minimum distance decoding. IEEE Trans. Inform. Theory **IT-12** (1966) 125–131

# Weaknesses in Shared RSA Key Generation Protocols

l k \* l k l \*\* k  
l \*\*\*

Department of Mathematics,  
Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, United Kingdom.  
{S.Blackburn,M.Burmester,S.Galbraith}@rhnc.ac.uk,  
sblakewi@certicom.com

**Abstract.** Cocks proposed a protocol for two parties to jointly generate a shared RSA key. His protocol was designed under the assumption that both parties follow the protocol. Cocks proposed a modification to the protocol to prevent certain attacks by an active adversary. The paper presents attacks that show that the Cocks protocols are not secure when one party deviates from the protocol.

## 1 Introduction

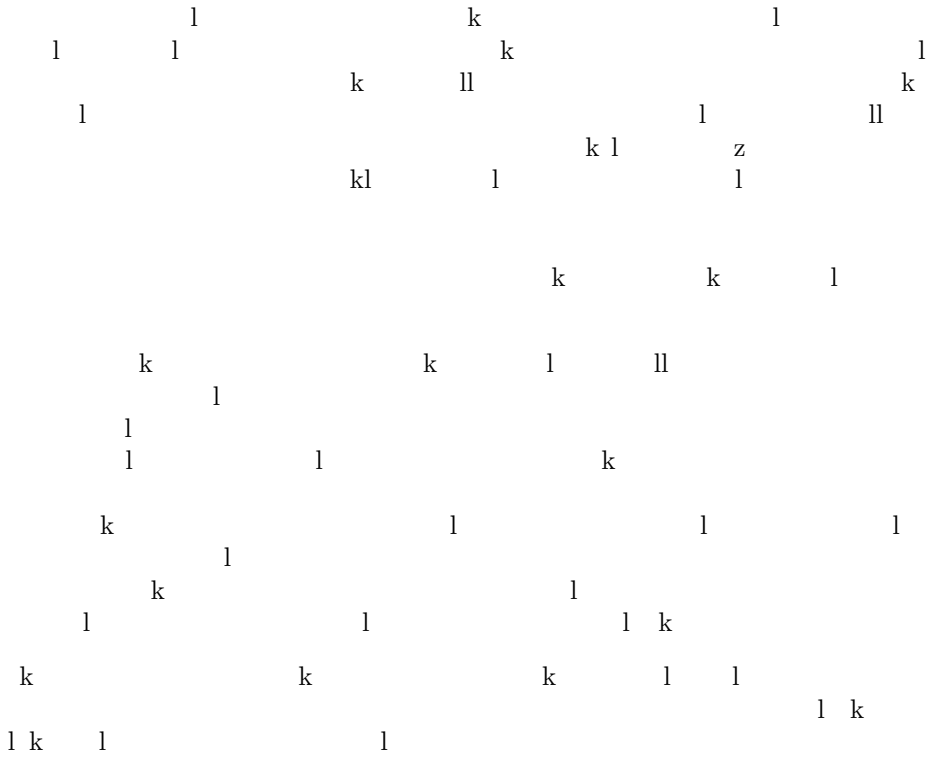
l l l l l  
k k k k k  
l l l l l  
k k k k k  
ll l l l l l  
9 l l l l l  
l l l l l  
l l l l l  
l l l l l  
k k k k k  
l l l l l  
k k k k k  
l l l l l  
kl kl kl kl kl  
l l l l l

---

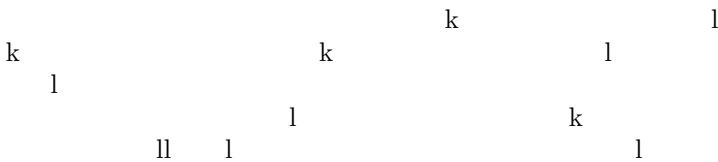
\* The author is supported by an EPSRC Advanced Fellowship.

\*\* The author is an EPSRC CASE student sponsored by Racal Airtech.

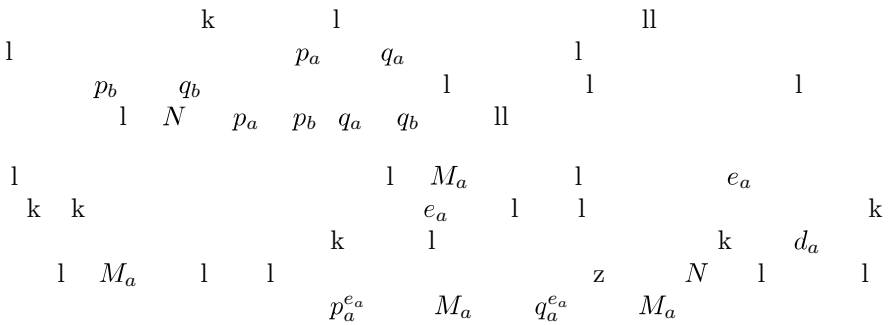
\*\*\* The author thanks the EPSRC for support.



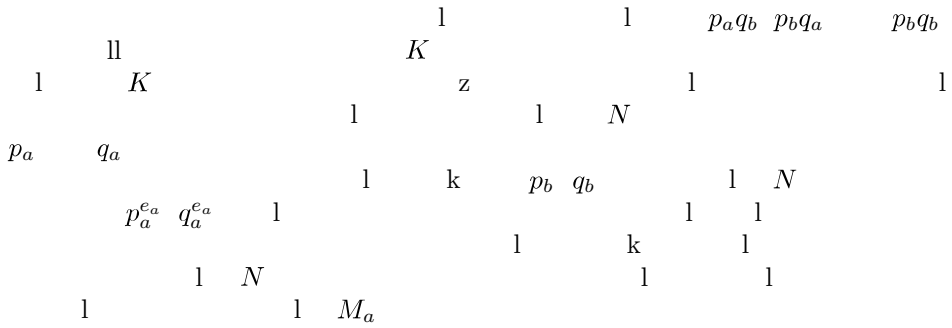
## 2 The Cocks Protocols



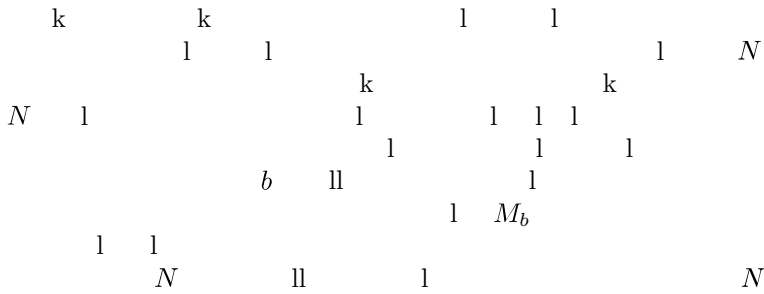
### 2.1 The Asymmetric Cocks Protocol



$$\begin{aligned} & \begin{matrix} 1 & 1 & & 1 \\ & & & \\ & a_{1,a} & p_a^{e_a} q_b^{e_a} & p_a q_b^{e_a} & M_a \\ & a_{2,a} & p_b^{e_a} q_a^{e_a} & p_b q_a^{e_a} & M_a \\ & a_{3,a} & p_b q_b^{e_a} & & M_a \end{matrix} \\ & \begin{matrix} & K & & b_{i,j,a} & i & , & j & K \\ 1 & M_a & & & & & & \end{matrix} \\ & \sum_{j=1}^K b_{i,j,a} M_a \parallel i \quad , \quad . \\ & \begin{matrix} K & & 1 & 1 & & & K \\ & 1 & 1 & & K & & x_{i,j,a} & a_{i,a} b_{i,j,a}^{e_a} & M_a \\ 1 & & & & 1 & & 1 & & \\ & & 1 & & 1 & & i,j & & \\ 1 & & 1 & 1 & 1 & & x_{i,j,a} & y_{i,j,a} & x_{i,j,a}^{d_a} & M_a \\ & & & & & & y_{i,j,a} & b_{i,j,a} & a_{i,a}^{d_a} & M_a \end{matrix} \\ & \begin{matrix} 1 \\ p_a q_a \sum_{i=1}^3 \sum_{j=1}^K y_{i,j,a} & p_a q_a & p_a q_b & p_b q_a & p_b q_b & N & M_a \end{matrix} \\ & \begin{matrix} < N < M_a & 1 & N & 1 \\ 1 & N & & & \end{matrix} \\ & \begin{matrix} N & 1 & 1 & 1 & & N & \\ & & & 1 & & & \text{kl} \\ & & & 1 & & N & \end{matrix} \\ & \begin{matrix} \parallel & 1 & & \parallel & 1 & e & 1 \\ d_a & d_b & & d & & & p_a & q_a \\ p_b & q_b & 1 & e & & \text{kl} & & \end{matrix} \\ & \text{Security.} \\ & \begin{matrix} & 1 & & 1 \\ & 1 & 1 & 1 & K & 1 & & K \\ 1 & 1 & & & & & \sum_{i=1}^1 \sum_{j=1}^K y_{i,j,a} & \parallel \\ & & p_a q_b & p_b q_a & p_b q_b & k & & 1 \end{matrix} \\ & \frac{K}{K^3} > M_a^2 \quad , \end{aligned}$$



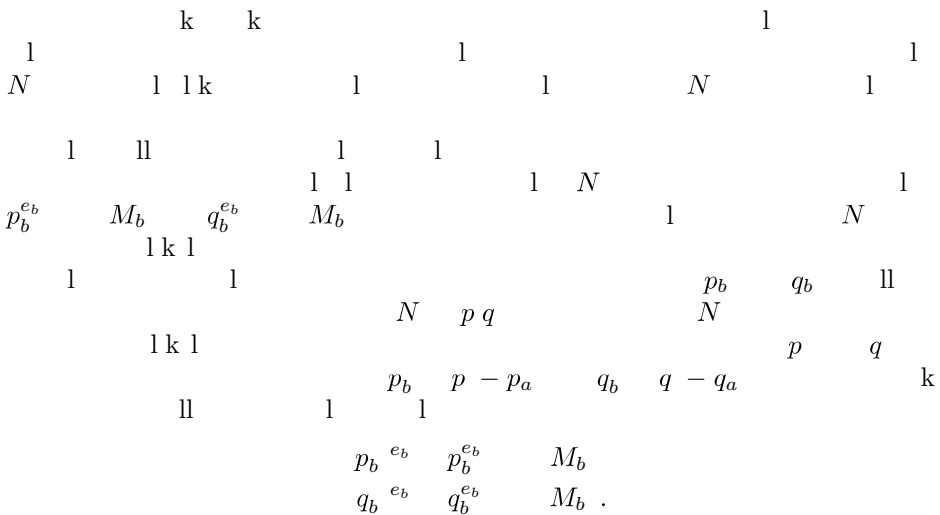
## 2.2 The Symmetric Protocol



## 3 Attacks on the Cocks Protocols



### 3.1 Dishonest Alice in the Symmetric Protocol



$$-y_{i,j,b} - \mathbb{Z}/M_b\mathbb{Z} \quad -i - \quad , \quad -j - K -$$

$$\sum_{i=1}^3 \sum_{j=1}^K y_{i,j,b} \quad N - p_b q_b \quad M_b \quad .$$

### 3.2 Cheating during the Boneh-Franklin Test

[illegible]

3.3 Dishonest Bob in the Asymmetric Protocol

$$k \quad \begin{matrix} & 1 & & 1 & & x_{i,j,a} & & 1 \\ & & 1 & & y_{i,j,a} & & x_{i,j,a}^{d_a} & & 1 & & M_a & & 1 \\ & & & 1 & & y_{i,j,a} - \mathbb{Z}/M_a\mathbb{Z} & & & & 1 & & & & x_{i,j,a} \\ y_{i,j,a}^{e_a} & & M_a & & 1 & & 1 & & 1 & & & & \end{matrix}$$

$$N \quad p_a q_a \quad \sum_{i=1}^3 \sum_{j=1}^K x_{i,j,a}^{d_a} \quad M_a$$

$$\begin{matrix} & & 1 & & 1 & & 1 & & p_a q_a \\ & & & 1 & & & & & 1 \\ p_a & & q_a & & k & & 1 & & & & 1 & & 1 \\ & & & 1 & & & kl & & & & 1 \\ & & & & 1 & & & & N & & & & \\ & & & & & k & & 1 & & & & & p_a^{e_a} \\ M_a & & q_a^{e_a} & & M_a & & 1 & & & & 1 & & 1 \\ & p_b & p_a^{c_1} & q_a^{c_2} & c_3 & & c_1 & c_2 & c_3 & & & & 1 & 1 \\ & q_b & 1 & & & & k & & p_a & & q_a & & 1 & & 1 & & 1 & & p_b^{e_a} \\ q_b^{e_a} & & 1 & & M_a & & & & & & 1 & & & & & & & & \\ 1 & 1 & & 1 & & & x_{i,j,a} & & ll & & N & & & & & & & & \\ & & 1 & & & k & p_b & q_a & q_b & p_a & & 1 & & N & & & & & \\ & & & 1 & & & 1 & & & & & 1 & & N & & & & & \\ & & & & & & & N & & & & & & & & & & & \end{matrix}$$

3.4 Cheating by a Choice of  $p_a, q_a$

$$\begin{matrix} & & & 1 & & 1 & & 1 \\ p_a & & q_a & & & & & 1 & & 1 & & M_a \\ & & & & 1 & & & & & & & \\ 1 & & x_{i,j,a} & & 1 & & -x_{1,j,a} - & -x_{2,j,a} - & -x_{3,j,a} - \\ & & & & & & x_{i,j,a} & M_a & ll \\ & & & & 1 & & 1 & & N & & & \\ & & & & & 1 & & k & & & & 1 \\ & & & k & & 1 & & 1 & & k & & \\ & & 1 & & & 1 & & & 1 & M_a & & M_b \end{matrix}$$

4 Acknowledgements

$$\begin{matrix} & & 1 & & 1 & & k \\ & & & & & 1 & & z & & ll & & 1 \\ 1 & k & & k & & & & & 1 & ll & & \end{matrix}$$

## References

1. M. Bellare and S. Goldwasser, *Lecture Notes in Cryptography*. 1996. Available at <http://www-cse.ucsd.edu/users/mihir/>
2. S.R. Blackburn, S. Blake-Wilson, M. Burmester and S.D. Galbraith, 'Shared generation of shared RSA keys' Technical report CORR 98-19, University of Waterloo.  
Available from <http://www.cacr.math.uwaterloo.ca/>
3. D. Boneh and M. Franklin, 'Efficient generation of shared RSA keys', in B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97, Lecture Notes in Computer Science Vol. 1294*, Springer-Verlag, 1997, pp. 425–439.
4. C. Cocks, 'Split knowledge generation of RSA parameters', in M. Darnell, editor, *Cryptography and Coding: 6th IMA Conference, Lecture Notes in Computer Science Volume 1355*, Springer-Verlag, 1997, pp. 89–95.
5. C. Cocks, 'Split generation of RSA parameters with multiple participants', 1998. Available at <http://www.cesg.gov.uk>
6. D.E. Denning and D.K. Branstad, 'A taxonomy of key escrow encryption schemes', *Communications of the A.C.M.*, Vol. 39, No. 1 (1996), pp. 24–40.
7. A. Fiat and A. Shamir, 'How to prove yourself: Practical solutions to identification and signature problems', in A.M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science Vol. 263*, Springer-Verlag, 1987, pp. 186–194.
8. Y. Frankel, P.D. MacKenzie, M. Yung, 'Robust efficient distributed RSA key generation', In *Proc. of 30th STOC*, 1998, pp. 663–672.
9. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, 'Robust and efficient sharing of RSA functions', in N. Kobitz, editor, *Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science 1109*, Springer-Verlag, 1996, pp. 157–172.
10. N. Gilboa, 'Two party RSA key generation', in M. Weiner, editor, *Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science 1666*, Springer-Verlag 1999, pp. 116–129.
11. G. Poupard, J. Stern, 'Generation of shared RSA keys by two parties', In *ASIACRYPT '98*, 1998, pp. 357–371.

# Digital Signature with Message Recovery and Authenticated Encryption (Signcryption) - A Comparison

Chan Yeob Yeun<sup>\*</sup>

Information Security Group  
Royal Holloway, University of London  
Egham, Surrey TW20 0EX, UK  
`c.yeun@rhnc.ac.uk`

**Abstract.** Mitchell and Yeun [8] showed that Chen's scheme [2] is not a digital signature scheme with message recovery, whereas it should be called an authenticated encryption scheme. Also note that similar remarks have been made in [10] regarding schemes recently proposed by Zheng. Thus we will show that there are major differences between a digital signature scheme with message recovery and authenticated encryption scheme by proposing a digital signature with message recovery scheme and signcryption scheme as an example for comparison. The security of the schemes is based on intractability of solving the Diffie Hellman problem as well as finding a collision on one-way hash-function.

## 1 Introduction

In 1976, Diffie and Hellman [3] introduced the public-key cryptosystem which is based on the discrete logarithm problem (DLP). The intractability of the DLP is equivalent to the security of the ElGamal public-key scheme [4] and its digital signature scheme.

Among the current known signature schemes, RSA [11] is unique in the sense that the signature and encryption functions are inverse to each other. For this reason, an RSA signature can be used with message recovery. On the other hand, a discrete logarithm based signature, such as ElGamal [4] and DSS [1], cannot provide message recovery. The benefits of the message recovery are applications without a hash function and smaller bandwidth for signatures. In 1993, Nyberg and Rueppel [9] proposed the first digital signature with message recovery based on the discrete logarithm problem.

In many applications it is necessary to provide both confidentiality and integrity/origin protection for a transmitted message. This can be achieved using a combination of encryption and a digital signature. However, this doubles the cost of protection, and motivates the work of Horster, Michels and Petersen

---

<sup>\*</sup> The author is supported by a Research Studentship and Maintenance Award from RHBNC.

[5] who introduced an authenticated encryption scheme, designed to provide a combination of services at reduced cost.

Subsequently Lee and Chang [6] modified the HMP scheme to remove the need for a one-way function, whilst keeping communication costs the same. This scheme may be advantageous in environments where implementing a one-way function is difficult, e.g. in a smart card with limited memory and/or computational capability.

More recently, Chen [2] introduced a variant of the Lee and Chang scheme, which is claimed to provide the same security level with a simpler specification. However, some of the claims made by Chen are incorrect as Mitchell and Yeun [8] pointed out that Chen's scheme [2] is not a digital signature scheme with message recovery, whereas it should be called an authenticated encryption scheme. Also note that similar remarks have been made in [10] regarding schemes recently proposed by Zheng.

## 2 Comparison for a Digital Signature with Message Recovery and a Signcryption

We observe that there are major differences between a digital signature scheme with message recovery (see [9]) and authenticated encryption schemes (see [5,6]) as follows:

### 2.1 A Digital Signature with Message Recovery

Basically, a digital signature with message recovery scheme should satisfy the following properties.

- **Data integrity/origin protection:** This is property whereby data has not been altered in an unauthorised manner since the time it was created, transmitted, or stored by an authorised source as well as protecting one's origin.
- **Nonrepudiation:** It is computationally feasible for the TTP to settle a dispute between the signer and the recipient in an event where the signer denies the fact that he/she is the sender of the signed text to the recipient. To compare with an authenticated encryption (signcryption), the signer does not reveal any his/her private keys to the TTP.

Thus, in a digital signature with message recovery scheme, the trusted third party (TTP) can always verify the signatures which are sent by the receiver  $B$  without  $B$  having to divulge any long term secret information to the TTP.

### 2.2 An Authenticated Encryption (Signcryption)

Basically, an authenticated encryption (signcryption) scheme should satisfy the following properties.

- **Confidentiality:** It is computationally infeasible for an adaptive attacker to find out any secret information from signcrypted text.
- **Data integrity/origin protection:** It is computationally infeasible for an adaptive attacker to masquerade as the signcrypter in creating a signcrypted text as well as protecting one's origin.
- **Nonrepudiation:** It is computationally feasible for the TTP to settle a dispute between the signcrypter and the recipient in an event where the signcrypter denies the fact that he/she is the sender of the signcrypted text to the recipient. To compare with a digital signature with message recovery, the signcrypter reveal any his/her private keys to the TTP.

Thus, in a authenticated encryption schemes, only the sender  $A$  and the receiver  $B$  can only verify a protected message sent from  $A$  to  $B$ . This is because  $B$  can only verify such a message with the aid of his private decryption key. Therefore, one can deduces that this is an unacceptable property for a signature scheme as discussed in section 2.1, where one would normally expect signature verification to be possible without compromise of any private keys.

In the following, we will propose a digital signature with message recovery scheme and signcryption scheme as an example for comparison. The security of the systems is related to the security of Diffie-Hellman [3] and that of randomly chosen one-way collision resistance hash-function. Assume that Diffie-Hellman and one-way collision resistance hash-function are easy to break, then so is the proposed schemes.

### 3 System Generation

The key centre selects and publishes the system parameters for public usage. Let  $p$  be a prime with  $2^{511} < p < 2^{512}$ ,  $q$  a prime divisor of  $p-1$  with  $2^{159} < q < 2^{160}$ ,  $g_1$  and  $g_2$  ( $1 < g_1, g_2 < p$ ) integers of order  $q$  and  $R$  a redundancy function (see Section 11.2.3 of [7]), and its inverse  $R^{-1}$ , and  $h$  is one-way collision resistant hash-function (see Section 9.2.2 of [7]).  $p, q, g_1, g_2, R, R^{-1}$  and  $h$  are publicly known.

Suppose Alice has two private keys  $X_{A_1}, X_{A_2}$  ( $1 < X_{A_1}, X_{A_2} < q$ ), and two public keys:

$$P_{A_1} = g_1^{X_{A_1}} \bmod p, P_{A_2} = g_2^{X_{A_2}} \bmod p.$$

Similarly, suppose Bob has two private keys  $X_{B_1}, X_{B_2}$ , ( $1 < X_{B_1}, X_{B_2} < q$ ), and two public keys:

$$P_{B_1} = g_1^{X_{B_1}} \bmod p, P_{B_2} = g_2^{X_{B_2}} \bmod p.$$

In addition, every participant must have a means of obtaining a verified copy of every other participant's public signature verification keys. This could, for example, be provided by having the key centre certify every participant's public keys, and having every participant distribute their certificate with every signed message they send.

## 4 A Digital Signature with Message Recovery Scheme

To sign a message  $m \in \mathbb{Z}_p$ , Alice randomly chooses two integers  $k_1$  and  $k_2$ ,  $1 < k_1, k_2 < q$  and computes the following:

$$m = R(m),$$

$$r = m g_1^{-k_1} g_2^{-k_2} \bmod p,$$

$$s_1 = k_1 - h(r)X_{A_1} \bmod q,$$

and

$$s_2 = k_2 - h(r)X_{A_2} \bmod q.$$

Then Alice sends  $\text{Sig}(m) = (r, s_1, s_2)$  to Bob. After receiving  $\text{Sig}(m)$ , the message can be recovered by Bob as follows:

$$m = r g_1^{s_1} g_2^{s_2} P_{A_1}^{h(r)} P_{A_2}^{h(r)} \bmod p.$$

After checking the validity of  $m$ , the message can be recovered by computing

$$m = R^{-1}(m).$$

This digital signature is secure if one selects a secure redundancy function  $R$  and a randomly chosen one-way collision-resistant hash-function  $h$  are used and providing that solving two discrete logarithms problems are computationally infeasible.

Observe that this scheme is a digital signature with message recovery as discussed in section 2.1, i.e. it satisfies the data integrity/ origin protection and nonrepudiation. The trusted third party (TTP) can always verify the signatures which are sent by the receiver Bob without Bob having to divulge two long term private keys to the TTP.

## 5 An Authenticated Encryption (Signcryption) Scheme

Suppose that Alice wants to send a message  $m$  to Bob. Then she first chooses two random integers  $k_1$  and  $k_2$ ,  $1 < k_1, k_2 < q$  and computes the following:

$$K_1 = (P_{B_1}^{k_1} \bmod p) \bmod q,$$

$$K_2 = (P_{B_2}^{k_2} \bmod p) \bmod q,$$

$$m = R(m),$$

$$r = m K_1 - K_2 \bmod p,$$

$$s_1 = k_1 - h(r)X_{A_1} \bmod q$$

and

$$s_2 = k_2 - h(r)X_{A_2} \bmod q.$$

Then Alice sends  $(r, s_1, s_2)$  to Bob. After receiving  $(r, s_1, s_2)$ , Bob computes the following:

$$P_{A_1B_1} = g_1^{X_{A_1}X_{B_1}} \bmod p,$$

$$P_{A_2B_2} = g_2^{X_{A_2}X_{B_2}} \bmod p,$$

$$K_1 = (P_{B_1}^{s_1} P_{A_1B_1}^{h(r)} \bmod p) \bmod q$$

and

$$K_2 = (P_{B_2}^{s_2} P_{A_2B_2}^{h(r)} \bmod p) \bmod q.$$

Thus, he computes

$$K_1^{-1}(r + K_2) \bmod p = m \bmod p.$$

After checking the validity of  $m$ , the message can be recovered by computing

$$m = R^{-1}(m).$$

This authenticated encryption (signcryption) scheme is secure if one chooses a secure redundancy function  $R$  and a randomly chosen one-way collision-resistant hash-function is used and providing that solving the discrete logarithms are computationally infeasible.

Observe that this scheme is a authenticated encryption (signcryption) scheme as discussed in section 2.2, i.e. it satisfies confidentiality, data integrity/origin protection and nonrepudiation. Only the sender Alice and receiver Bob can verify an authenticated encryption message sent from Alice to Bob. This is because Bob requires his private keys for verification.

## 6 Conclusion

We have shown that there are major differences between a digital signature scheme with message recovery and authenticated encryption scheme. We also have proposed a new digital signature with message recovery scheme which satisfies the properties of data integrity/origin protection and nonrepudiation, and a new signcryption scheme which satisfies the properties of confidentiality, data integrity/origin protection and nonrepudiation are an example for comparison. The security of these schemes is based on intractability of solving the Diffie Hellman problem as well as finding a collision on one-way hash-function.

## 7 Acknowledgements

The author is grateful to Fred Piper for his support, and to Chris Mitchell and Mike Burmester for comments on an early draft of the paper.

## References

1. The digital signature standard proposed by NIST. *Communications of the ACM*, 35(7):36–40, 1992.
2. K. Chen. Signatrue with message recovery. *Electronics Letters*, 34(20):1934, 1998.
3. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
4. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1976.
5. P. Horster, M. Michels, and H. Petersen. Authenticated encryption schemes with low communication costs. *Electronics Letters*, 30(15):1212–1213, 1994
6. W. Lee and C. Chang. Authenticated encryption scheme without using a one-way function. *Electronics Letters*, 31(19):1656–1657, 1995.
7. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. Handbook of Applied Cryptography, CRC Press, 1997
8. C.J. Mitchell and C.Y. Yeun. Comment–Signature scheme with message recovery. *Electronics Letters*, 35(3):217, 1999.
9. K. Nyberg and R.A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Advances in Cryptography – Proceedings of EUROCRYPT '94*, pages 175–190, Springer-Verlag, 1995.
10. H. Petersen and M. Michels. Cryptanalysis and improvement of signcryption schemes. *IEE Proceedings on Computers and Digital Techniques*, 145:149–151, 1998.
11. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

# Index

r r					
r		0		r	r
y					
r		00		r	
			00		
r					0
r	r	r		"	r
r				y	
y			0		
r					
r	y			r	
r	r	00			
		0			
	0			rr	
rr				r	
	0				
	0			y	0
r		0		r	
r		00		r	
r				r	
	0			y	0
			0		
			r	r	
ry					
			r	r	
				y r	0
				0	
r y				r	